



# **ownCloud Administrators Manual**

***Release 6.0***

**The ownCloud developers**

April 21, 2014



## CONTENTS

<b>1</b>	<b>ownCloud 6.0 Admin Documentation</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	1
1.3	Configuration . . . . .	2
1.4	Apps . . . . .	2
1.5	Maintenance . . . . .	3
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Appliances . . . . .	5
2.2	Linux Distributions . . . . .	5
2.3	Mac OS X . . . . .	6
2.4	Windows 7 and Windows Server 2008 . . . . .	6
2.5	Univention Corporate Server . . . . .	12
2.6	Manual Installation . . . . .	19
2.7	PageKite Configuration . . . . .	30
2.8	Open Wrt . . . . .	30
<b>3</b>	<b>Configuration</b>	<b>31</b>
3.1	Managing Apps . . . . .	31
3.2	User Management . . . . .	33
3.3	LDAP Authentication . . . . .	36
3.4	Background Jobs . . . . .	48
3.5	Find Third-Party Libraries . . . . .	49
3.6	Automatic Configuration . . . . .	50
3.7	Customizing Client Download Links . . . . .	51
3.8	MySQL/Postgres/SQLite Support . . . . .	52
3.9	Using Server-Side Encryption . . . . .	57
3.10	Disable Knowledge Base . . . . .	58
3.11	Setting the Default Language . . . . .	59
3.12	Configure Logging . . . . .	59
3.13	Sending Mail Notifications . . . . .	60
3.14	Enable Maintenance Mode . . . . .	65
3.15	Enabling File Previews . . . . .	65
3.16	Reverse Proxy Configuration . . . . .	67
3.17	Dealing with Big File Uploads . . . . .	68
3.18	Custom Mount Configuration Web-GUI . . . . .	69
3.19	Custom Mount Configuration . . . . .	70
3.20	Custom User Backend Configuration . . . . .	75
3.21	Serving static files via web server . . . . .	77
<b>4</b>	<b>Apps</b>	<b>81</b>

4.1	Activity . . . . .	81
4.2	Finding Deployment Dependencies . . . . .	85
4.3	File Antivirus Engine . . . . .	87
4.4	Encryption . . . . .	90
4.5	External storage support . . . . .	94
4.6	Sharing . . . . .	108
4.7	Deleted Files . . . . .	113
4.8	Versions . . . . .	116
4.9	First Run Wizard . . . . .	120
4.10	LDAP user and group backend . . . . .	123
4.11	File Viewers . . . . .	131
<b>5</b>	<b>Maintenance</b>	<b>133</b>
5.1	Backing up ownCloud . . . . .	133
5.2	Updating ownCloud . . . . .	134
5.3	Restoring ownCloud . . . . .	135
5.4	Migrating ownCloud Installations . . . . .	135
<b>6</b>	<b>The Configuration File</b>	<b>137</b>
6.1	Default Parameters . . . . .	137
6.2	Reverse Proxy Configurations . . . . .	138
6.3	User Experience . . . . .	139
6.4	Mail Parameters . . . . .	140
6.5	Deleted Items . . . . .	141
6.6	Verification . . . . .	142
6.7	Logging . . . . .	142
6.8	Session Info . . . . .	143
6.9	Code Locations . . . . .	143
6.10	APPS . . . . .	144
6.11	Previews . . . . .	145
6.12	Maintenance . . . . .	145
6.13	Miscellaneous . . . . .	146
<b>7</b>	<b>Issues</b>	<b>147</b>
<b>8</b>	<b>Indices and tables</b>	<b>149</b>

## OWNCLOUD 6.0 ADMIN DOCUMENTATION

### 1.1 Introduction

This is the administrators manual for ownCloud, a flexible, open source file sync and share solution. It comprises of the ownCloud server, as well as client applications for Microsoft Windows, Mac OS X and Linux (Desktop Client) and mobile clients for the Android and Apple iOS operating system.

#### 1.1.1 Target audience

This guide is targeted towards people who want to install, administer and optimize ownCloud Server. If you want to learn how to use the Web UI, or how to install clients on the server, please refer to the [User Manual](#) or the [Desktop Client Manual](#) respectively.

#### 1.1.2 Structure of this document

The next chapters describes how to set up ownCloud Server on different platforms and operating systems, as well as how to update existing installations.

Further chapters will then detail on integrating ownCloud into your existing environment, e.g. how to setup LDAP or how to mount your storage.

### 1.2 Installation

This chapter will introduce you to the installation of ownCloud in different scenarios.

If you want to just try ownCloud in a virtual machine without any configuration, check the section [Appliances](#), where you will find ready-to-use images.

- *Linux Distributions* (recommended)
- *Windows 7 and Windows Server 2008*
- *Manual Installation*
- *PageKite Configuration*
- *Univention Corporate Server*
- *Mac OS X* (not supported)
- *Appliances*

## 1.3 Configuration

This chapter covers ownCloud and web server configuration.

- *Managing Apps*
- *User Management*
- *MySQL/Postgres/SQLite Support*
- *LDAP Authentication*
- *Custom Mount Configuration Web-GUI*
- *Custom Mount Configuration*
- *Background Jobs*
- *Sending Mail Notifications*
- *Automatic Configuration*
- *Using Server-Side Encryption*
- *Dealing with Big File Uploads*
- *Reverse Proxy Configuration*
- *Serving static files via web server*
- *Find Third-Party Libraries*
- *Custom User Backend Configuration*
- *Customizing Client Download Links*
- *Enable Maintenance Mode*
- *Disable Knowledge Base*
- *Configure Logging*
- *Setting the Default Language*

Finally, the chapter *The Configuration File* details on the switches in the `config.php` file.

## 1.4 Apps

This chapter covers individual ownCloud apps.

- *Activity*
- *Finding Deployment Dependencies*
- *File Antivirus Engine*
- *Encryption*
- *External storage support*
- *Sharing*
- *Deleted Files*
- *Versions*
- *First Run Wizard*

- *LDAP user and group backend*
- *File Viewers*

## 1.5 Maintenance

This chapter covers maintenance tasks such as updating or migrating to a new version.

- *Migrating ownCloud Installations*
- *Updating ownCloud*





## 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:** There are two packages for ownCloud: [stable version](#) in the official community repository and [development version](#) in AUR.

**PCLinuxOS:** Follow the Tutorial [ownCloud, installation and setup](#) on the PCLinuxOS web site.

## 2.3 Mac OS X

---

**Note:** Due to an [issue](#) with Mac OS Unicode support, installing ownCloud Server 6.0 on Mac OS is currently not supported.

---

## 2.4 Windows 7 and Windows Server 2008

---

**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).

It assumes that you have a vanilla, non-IIS enabled Windows machine – Windows 7 or Server 2008. After enabling IIS, the steps are essentially identical for Windows 7 and Windows Server 2008.

For installing ownCloud physical access or a remote desktop connection is required. You should leverage MySQL as the backend database for ownCloud. If you do not want to use MySQL, it is possible to use Postgres or SQLite instead. Microsoft SQL Server is not yet supported.

Enabling SSL is not yet covered by this section.

---

**Note:** If you make your desktop machine or server available outside of your LAN, you must maintain it. Monitor the logs, manage the access, apply patches to avoid compromising the system at large.

---

There are 4 primary steps to the installation, and then a 5th step required for configuring everything to allow files larger than the default 2MB.

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

1. Go to *Start -> Control Panel -> Programs*.
2. Under Programs and Features, there is a link titled *Turn Windows Features on and Off*. Click on it.
3. There is a box labeled Internet Information Services, expand it.

4. Expand World Wide Web Services and all the folders underneath.
5. Select the folders as illustrated in the picture below to get your IIS server up and running.

You do not need an FTP server running, so you should tune that feature off for your server. You definitely need the IIS Management Console, as that is the easiest way to start, stop, restart your server, as well as where you change certificate options and manage items like file upload size. You must check the CGI box under Application Development Features, because CGI is how you enable PHP on IIS.

You have to turn off WebDAV publishing or the Windows WebDAV conflicts with the ownCloud WebDAV interface. This might already be turned off for you, just make sure it stays that way. The common HTTP features are the features you would expect from a web server. With the selections on this page, IIS will now serve up a web page for you.

Restart IIS by going to the IIS manager (*Start → IIS Manager*).

Select your website, and on the far right side is a section titled *Manage Server*. Make sure that the service is started, or click *Start* to start the services selected. Once this is complete, you should be able to go to a web browser and navigate to <http://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.

## Windows Server 2008

1. Go to *Start → Control Panel → Programs*.
2. Under Programs and Features, there is link titled *Turn Windows Features on and Off*. Click on it.
3. This will bring up the Server Manager.
4. In the server manager, Click on Roles, and then click Add Roles.
5. Use the *Add Roles Wizard* to add the web server role.
6. Make sure that, at a minimum, the same boxes are checked in this wizard that are checked in the Windows 7 Section. For example, make sure that the CGI box is checked under Application Development Features, and that WebDAV Publishing is turned off. With Remote Desktop Sharing turned on, the detailed role service list looks like the figure “Role Services”.
7. Restart IIS by going to the IIS manager (*Start → IIS Manager*).
8. Select your website, and on the far right side is a section titled *Manage server*. Make sure that the service is started, or click “Start” to start the services selected.
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. The next part of this “how to” installs PHP on the server.

### 2.4.2 Installing PHP

This part is also straightforward, but it is necessary to remind you that this is for IIS only.

1. Go to the following link and grab the [PHP installer](#) for version “VC9 Non Thread Safe” 32 or 64 bit based on your system.

---

**Note:** If you are using Apache, make sure you grab VC6 instead, lower on the page.

---

2. Once through that login, select the location that is closest to you geographically.

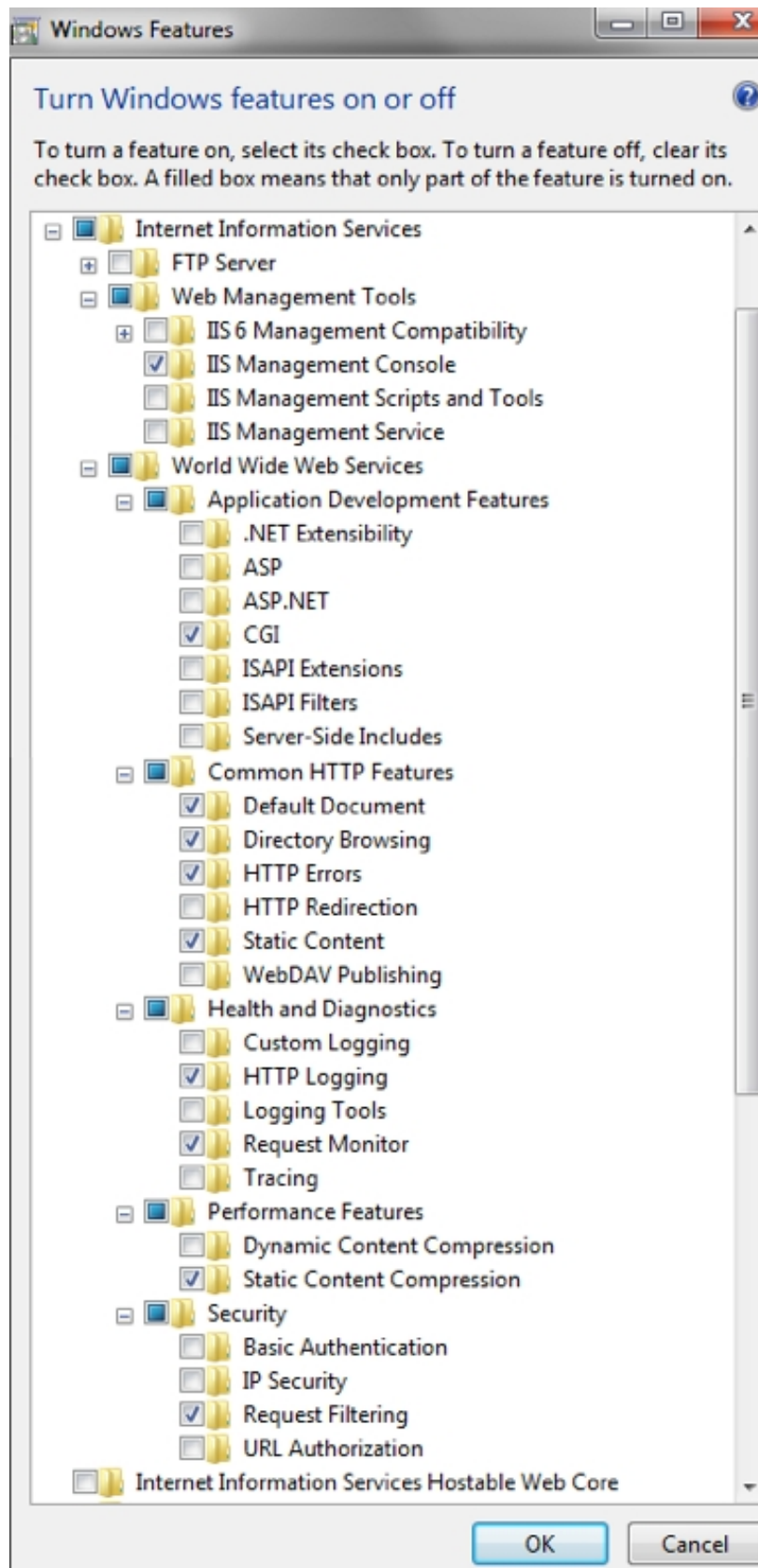


Figure 2.1: Windows Features required for ownCloud on Windows 7

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 7 and Windows Server 2008	Installed
IIS 6 Management Console	Installed
FTP Server	Not installed

3. Run that install wizard once it is downloaded. Read the license agreement, agree, select an install directory.
4. Then select IIS FastCGI as the install server.
5. Take the default selections for the items to install, and click next. Then click *install*.
6. After a few minutes, PHP will be installed. On to MySQL.

### 2.4.3 Installing MySQL

This part installs MySQL on your Windows machine.

1. Point your browser to <http://dev.mysql.com/downloads/> and download the latest community edition for your OS – the 32 or 64 bit version. Please download the **MSI Installer** as it will make life easier.
2. Once downloaded, install MySQL (5.5 at the time of writing). Select the Typical installation.
3. When that finishes, check the box to launch the MySQL Instance Configuration Wizard and click Finish.
4. Select a standard configuration, as this will be the only version of MySQL on this machine.
5. Select to install as a windows service, and Check the Launch the MySQL Server Automatically button.
6. Select the modify security settings box on the next page, and enter a password you will remember. You will need this password when you configure ownCloud.
7. Uncheck **enable** root access from remote machines” for security reasons.
8. Click execute, and wait while the instance is created and launched.
9. Click Finish when this is all complete.

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>.
2. It will arrive as a tar.bz2 file, and I recommend something like jZip for a free utility to unzip it.
3. Once you have the ownCloud directory unzipped and saved locally, copy it into your wwwroot directory (probably **C:\inetpub\wwwroot**).

---

**Note:** You cannot install directly into the directory **wwwroot** from jzip, as only the administrator can unzip into the **wwwroot** directory. If you save it in a different folder, and then move the files into **wwwroot** in windows explorer, it works. This will install ownCloud locally in your root web directory. You can use a subdirectory called owncloud, or whatever you want – the www root, or something else.

---

4. It is now time to give write access to the ownCloud directory to the ownCloud server: Navigate your windows explorer over to **inetpub\wwwroot\owncloud** (or your installation directory if you selected something different).
5. Right click and select properties. Click on the security tab, and click the button “to change permissions, click edit”.
6. Select the “users” user from the list, and check the box “write”.

7. Apply these settings and close out.
8. Now open your browser and go to <http://localhost/owncloud> (or localhost if it is installed in the root www directory). This should bring up the ownCloud configuration page.
9. At this page, you enter your desired ownCloud user name and password for the administrator, and expand the little arrow.
10. 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 for MySQL earlier in step 3, and pick any name for the database instance.

---

**Note:** The ownCloud admin password and the MySQL password CANNOT be the same in any way.

---

11. Click next, and ownCloud should have you logged in as the admin user, and you can get started exploring ownCloud, creating other users and more!

### 2.4.5 Ensure Proper HTTP-Verb handling

IIS must pass all HTTP and WebDAV verbs to the PHP/CGI handler, and must not try to handle them by itself. If it does, syncing with the Desktop and Mobile Clients will fail. Here is how to ensure your configuration is correct:

1. Open IIS Manager.
2. In the *Connections* bar, pick 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 click *PHP\_via\_fastCGI*.
4. Choose *Request Restrictions* and find the *Verbs* tab.
5. Ensure *All Verbs* is checked.
6. Click *OK*.
7. Next, choose *Request Filtering* feature from IIS Manager.
8. Ensure that all verbs are permitted (or none are forbidden) in the *Verbs* tab.

Also, ensure that you did not enable the WebDAV authoring module, since ownCloud needs to be able to handle WebDAV on the application level.

### 2.4.6 Configuring ownCloud, PHP and IIS for Large File Uploads

Before going too nuts on ownCloud, it is important to do a couple of configuration changes to make this a useful service for you. You will probably want to increase the **max upload size**, for example. The default upload is set to **2MB**, which is too small for even most MP3 files.

To do that, simply go into your **PHP.ini** file, which can be found in your **C:\Program Files (x86)\PHP** folder. In here, you will find a **PHP.ini** file. Open this in a text editor, and look for a few key attributes to change:

- **upload\_max\_filesize** – change this to something good, like 1G, and you will get to upload much larger files.
- **post\_max\_size** – also change this size, and make it larger than the max upload size you chose, like 1G.

There are other changes you can make, such as the timeout duration for uploads, but for now you should be all set in the **PHP.ini** file.

Now you have to go back to IIS manager and make one last change to enable file uploads on the web server larger than 30MB.



1. Go to the start menu, and type **iis manager**.
2. Open IIS Manager Select the website you want enable to accept large file uploads.
3. In the main window in the middle double click on the icon **Request filtering**.
4. Once the window is opened you will see a bunch of tabs across the top of the far right,  
Select *Edit Feature Settings* and modify the *Maximum allowed content length (bytes)*
5. In here, you can change this to up to 4.1 GB.

---

**Note:** This entry is in BYTES, not KB.

---

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	Wether a new user is allowed to use ownCloud by default.	2012.0.1
owncloud/group/enabled	0	Wether 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

---

Continued on next page



Table 2.1 – continued from previous page

Key	Default	Description	Introduced
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/UIDAttribute	(empty)	Attribute that provides a unique value for each user and group entry. Empty value for autodetection.	5.0.9
owncloud/ldap/loginFilter	(&(l(&(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	(&(l(&(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,mid,primaryAddress	Attributes that shall be used 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

Continued on next page

Table 2.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/ldap/group/searchAttributes	cn,description, mailPrimaryAddress	Attributes taken into consideration when searching for groups (comma separated)	5.0.9
owncloud/join/users/update	yes	Whether ownCloud LDAP schema should be applied to existing users	2012.0.1
owncloud/group/enableDomainUsers	1	Whether the group “Domain Users” shall be enabled for ownCloud on install	2012.4.0.4
owncloud/join/users/filter	(&(l(&(objectClass=posixAccount)(objectClass=shadowAccount)(objectClass=univentionMail)(objectClass=sambaSamAccount)(objectClass=simpleSecurityObject)(&(objectClass=person)(objectClass=organizationalPerson)(objectClass=inetOrgPerson)))(!!(uidNumber=0))(!(l(uid=*\$)(uid=owncloudsystemuser)(uid=join-backup)(uid=join-slave))))(!(objectClass=ownCloudUser)))	Filters, on which LDAP users the ownCloud schema should be applied to. The default excludes system users and already ownCloudUsers.	2012.0.1
owncloud/join/groups/filter	(empty)	Filters which LDAP groups will be en/disabled for ownCloud when running the script /usr/share/owncloud/update-groups.sh	2012.4.0.4

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

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

or via UMC:

## Univention Configuration Registry

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

Entries

Category

All

Search attribute

All

Keyword

owncloud\*

Search

<input type="checkbox"/> UCR variable	Value	Edit	Delete
<input type="checkbox"/> owncloud/db/name	owncloud		
<input type="checkbox"/> owncloud/directory/data	/var/lib/owncloud		
<input type="checkbox"/> 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)))		
<input type="checkbox"/> owncloud/join/users/update	yes		
<input type="checkbox"/> owncloud/ldap/displayName	uid		
	(&(!(&(objectClass=posixAccount)(objectClass=shadowAccount))		

0 entries of 10 selected

Add

### 2.5.2 Installation

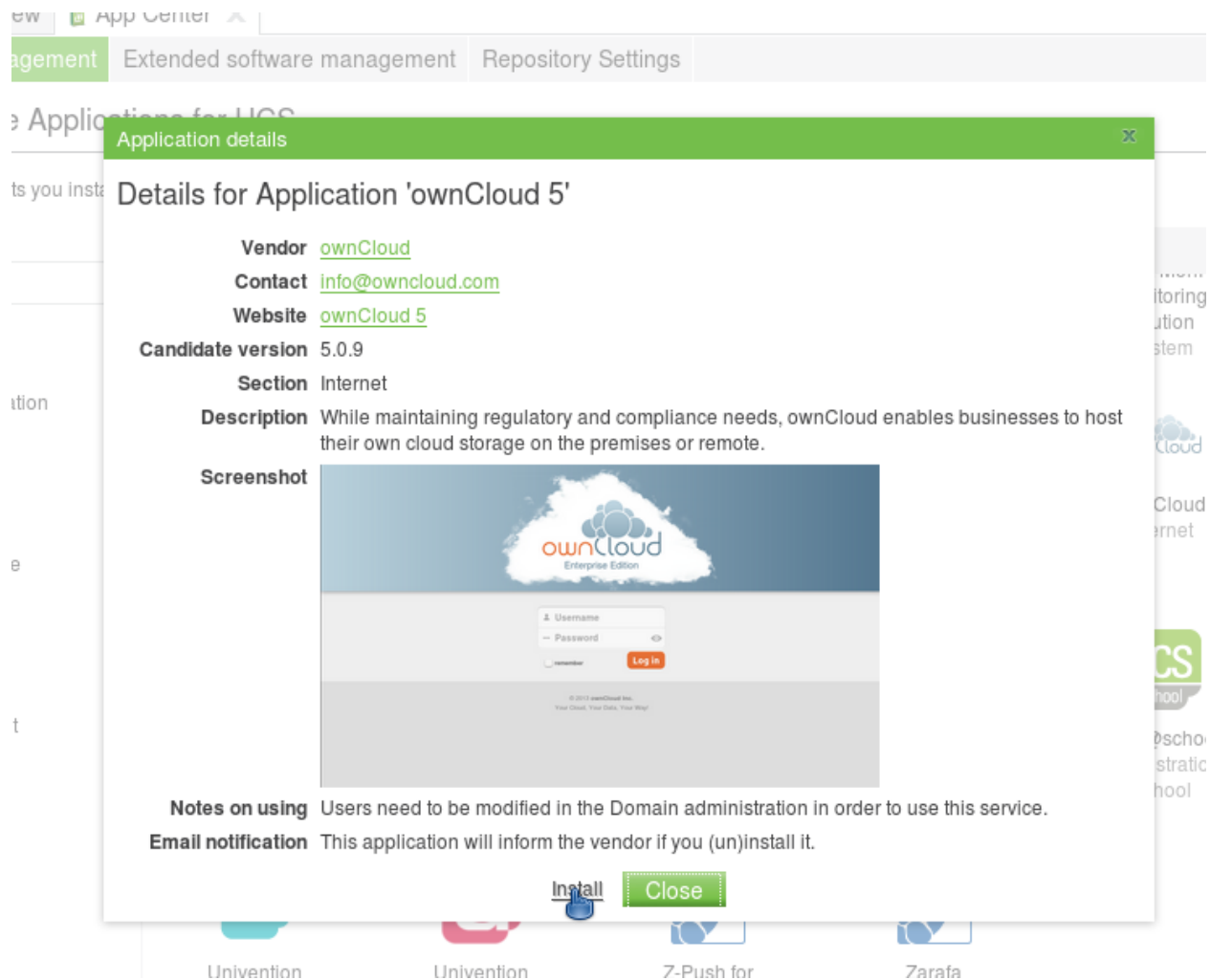
Now, we are ready to install ownCloud. This can be either done through the UCS App Center (recommended) or by downloading the packages.

#### UCS App Center

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

The screenshot displays the 'App Center' interface within the ownCloud administration panel. At the top, there are tabs for 'Overview' and 'App Center'. Below these are navigation links: 'App management', 'Extended software management', and 'Repository Settings'. The main heading is 'Manage Applications for UCS'. A descriptive text states: 'This page lets you install and remove applications that enhance your UCS installation.' On the left, there is a 'Search term' input field and a 'Categories' list including: All, Administration, CRM, DMS, Desktop, ERP, Groupware, Internet, Mail, School, System, Thin Client, and VoIP. The main area is titled 'Applications' and contains a grid of application tiles. Each tile shows an icon, the application name, and its category. The applications visible are: Groupware Webmail Edition (Groupware, Mail), Kivitendo ERP, KIXbox Administration, Kolab Groupware Solution (Groupware, Mail), Open-Xchange App Suite (Groupware, Mail), ownCloud Internet, ownCloud 5 Internet, RDP Server (xrdp) Administration, SRS Backup Administration, SugarCRM Community Edition CRM, UCS@school Administration, School, Univention Corporate Client (Desktop, Thin Client), Univention Demo App System, Z-Push for Zarafa Groupware, and Zarafa Collaboration Platform. A tooltip is visible over the 'RDP Server (xrdp)' application, stating: 'cloud solution for data and file sync, share and view'. At the bottom right, there is a link that says 'Suggest new app'.

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)

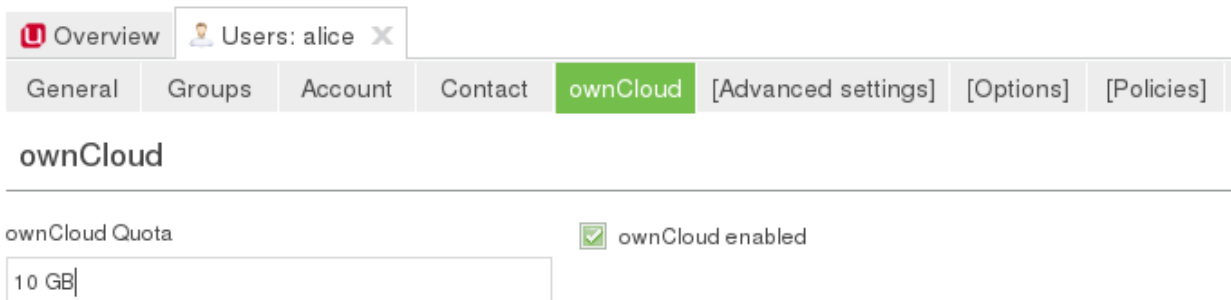
There is only one local admin user “owncloudadmin”, you can find his password in `/etc/owncloudadmin.secret`. Use this account, if you want to change basic ownCloud settings.

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.

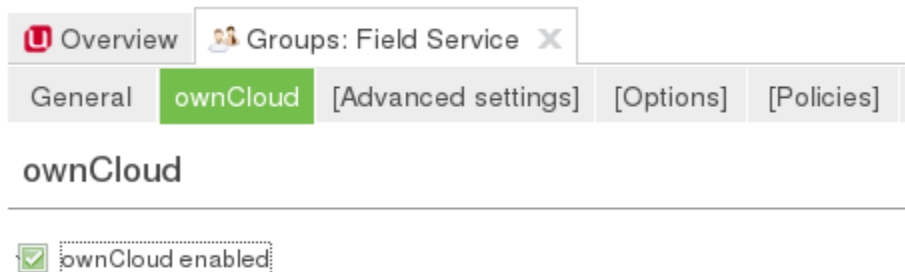
If you did not, go to the UMC and enable the users who shall have access (see picture below). Then, login at <http://myserver.com/owncloud/> with your LDAP credentials.



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 on 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.

Although this document tries to describe all aspects of setting up ownCloud, a basic understanding of the Linux operating system and of server administration is required.

### 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:

- PHP ( $\geq$  5.3.3 minimum, 5.4 or higher recommended)
- 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 (requires sqlite  $\geq$  3.0; simple configuration, but inferior performance)
- PHP module mysql
- PHP module pgsql (requires PostgreSQL  $\geq$  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)

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.

## Installation of packages 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).

## 2.6.2 Download, extract and copy ownCloud to Your Web Server

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 webserver (you can skip this step if you already downloaded and extracted the files there):

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

where /path/to/your/webserver/document-root, needs to be replaced by the actual path where the document root of your webserver is configured to be.

- If you don’t know where your webserver’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 webserver. For installing it anywhere else, you’ll have to adapt the above command (and also the commands in the following section) accordingly.

## 2.6.3 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 following command will change the ownership of the whole ownCloud folder to that user.

- The generic command to run is:

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

where `<php-user>` is to be replaced by the user running php scripts, and `/path/to/your/webserver/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 all Debian-based distributions (like Ubuntu, Debian or Linux Mint) and Gentoo, use `www-data` user
- On ArchLinux, use `http` user.
- On Fedora, use `apache` user.
- When you had extracted ownCloud as user `root`, you should adjust file and directory permission to avoid world writeable files and folder:

```
find /path/to/your/webserver/document-root/owncloud -type d -exec chmod 750 {} \;  
find /path/to/your/webserver/document-root/owncloud -type f -exec chmod 640 {} \;
```

Running this in combination with the above `chown` command will give a secure set-up.

## 2.6.4 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 the [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 ~ ^/(data|config|\.ht|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))(/.*)?$ {
    try_files $1 =404;

    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$1;
    fastcgi_param PATH_INFO $2;
    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;
}

}
```

To enable SSL support: - 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 changing
patch -p1 -i /patch/to/downloaded/patch.patch
quilt refresh
```

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

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

## 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 nonexistent 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.

### 2.6.5 Follow the Install 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>
  - 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
- Enter username and password for the administrative user account
- Expand Advanced options to choose a data folder and the database system
- If you are not using Apache as the web server, please set the data directory to a location outside of the document root.
- If following the Ubuntu-Apache-MySQL walk-through:
  - 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



- \* 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.
- In general, you have the following choices regarding the database:
  - For basic installs we recommend SQLite as it is easy to setup (ownCloud will do it for you). The performance when using sqlite is however inferior to the two other options.
  - For larger installs you should use MySQL or PostgreSQL.
  - Note that you will only be able to choose among the php database connectors which are actually installed on the system (see package requirements above).
  - Further, 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.
    - \* You can enter the name of an existing database and the username/password of a user with restricted permissions
      - 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.
- 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.

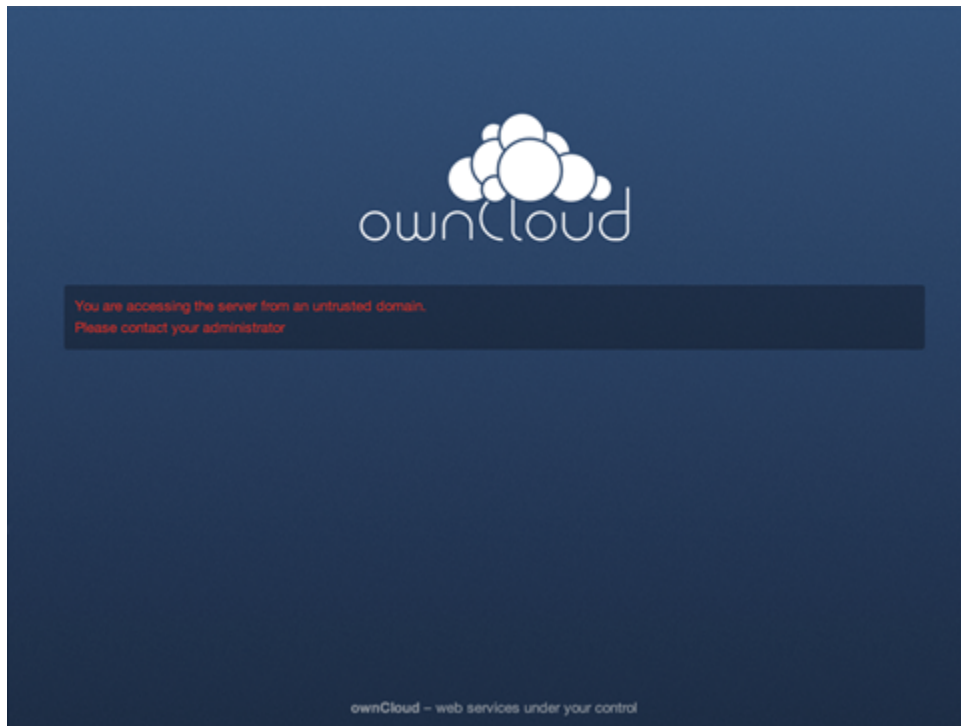
### 2.6.6 Note

When the initial ownCloud configuration is performed, ownCloud will take the URL used to access it and insert that the config.php file under the ‘trusted\_domains’ header.

Users will only be able to log into ownCloud when the addressed URL is as stated in the ‘trusted\_domains’ header in the config.php file.

In the event that a load balancer is in place, as long as it sends the correct X-Forwarded-Host header, there will be no issues.

It should be noted that the loopback address, 127.0.0.1, is white labeled and therefore users on the ownCloud server who access ownCloud with the loopback will successfully login. In the event that an improper URL is used, the following error will appear:



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

## 2.7 PageKite Configuration

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

## 2.8 Open Wrt

Here you can find a [tutorial for open Wrt](#)

## CONFIGURATION

### 3.1 Managing Apps

After you have installed ownCloud, you might realize that it would be nice to provide an additional function on top of the core functionality in your ownCloud installation.

With ownCloud installation, you will find some apps enabled by default. To see which applications are enabled, click on Apps button on the web interface navigation to go into applications page:

The screenshot shows the 'Administrator application page' in ownCloud. On the left is a sidebar with a list of navigation items: 'Share Files', 'Text Editor', 'Updater', 'Versions', 'Video Viewer', 'App Framework' (highlighted with a '3rd Party' badge), 'Encryption', 'External user support', 'LDAP user and group backend', 'ownCloud dependencies info', 'Tasks', 'WebDAV user backend', and 'More Apps ...'. The main content area displays the 'Encryption' app, version '0.5', marked as an 'Internal App'. The description states: 'The new ownCloud 5 files encryption system. After the app was enabled you need to re-login to initialize your encryption keys.' Below this, it mentions 'AGPL-licensed by Sam Tuke, Bjoern Schiessle, Florin Peter' and features an 'Enable' button.

#### Administrator application page

In this page, you can enable or disable applications simply by clicking on their names. Enabled applications will be shown in **bold** while disabled ones will be shown in normal font. If the app is not developed by ownCloud, it will have the *3rd party* notice next to it. To see what an application does, clicking on its name will show a description on the right side of the same page.

To install new apps, you can use *More apps* button or check out the [ownCloud apps store](#). There you will find a lot of ready-to-use apps provided by the ownCloud community.

If you would like to add your own app, please use *Add your App...* button on the same page. This will redirect you to our [Developer Center](#).

### 3.1.1 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,
    ),
),
```

#### Use your own appstore

If you want to allow the installation of apps from the apps store you have to set **appstoreenabled** parameter, but this can only be done if at least one of the configured apps directories is writable.

The **appstoreurl** is used to set the http path to the ownCloud apps store. The appstore server has to use OCS (Open Collaboration Services).

```
<?php
```

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

#### Guard against malicious 3rdparty code





Finally you can enable checks for malicious code fragments of 3rd-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:

Login Name	Password	Groups ▼	Create	Default Storage	1 GB ▼
Username	Full Name	Password	Groups	Group Admin	Storage
 admin	admin	••••••	admin ▼	Group Admin ▼	Default ▼
 test	test	••••••	test ▼	test ▼	1 GB ▼
 test2	test2	••••••	admin, test ▼	Group Admin ▼	Default ▼
 test3	test3	••••••	admin ▼	Group Admin ▼	512 MB ▼

### Users management page

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 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.

**Storage** 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 **Storage** 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 Storage** field in the action bar, 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. Open the **Groups** drop-down list in the action bar.

2. Click **add group**.
3. 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 edited (e.g. renamed) or removed. This feature will be available in a future version of ownCloud.

---

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

---

## 3.3 LDAP Authentication

ownCloud ships an LDAP backend, which allows full use of ownCloud for users 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 configuration follows a wizard-like approach, split into four tabs. A correctly completed first tab ("Server") is mandatory to access the other tabs. The settings in the other tabs are detected automatically, but should be reviewed by the admin. 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 Tab 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.



The screenshot shows the 'Server' configuration tab in the ownCloud administration interface. It includes a dropdown for '1. Server:', a 'Delete Configuration' button, and several input fields: 'Host', 'Port', 'User DN', 'Password', and 'One Base DN per line'. At the bottom, there is a 'Configuration incomplete' status message and buttons for 'Continue' and 'Help'.

**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. The port will be detected automatically, if the LDAP server is running on a standard port. 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 DN's 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 important that the backup server is a replica of the main server, because the object UUIDs must match.

The screenshot shows the 'Advanced' tab of the ownCloud LDAP configuration interface. Under the 'Connection Settings' section, the following options are visible:

- Configuration Active:** Checked (indicated by a red checkmark icon).
- Backup (Replica) Host:** An empty text input field.
- Backup (Replica) Port:** An empty text input field.
- Disable Main Server:** An unchecked checkbox.
- Case insensitive LDAP server (Windows):** An unchecked checkbox.
- Turn off SSL certificate validation:** An unchecked checkbox.
- Cache Time-To-Live:** A text input field containing the value '600'.

Below the Connection Settings section, there are two expandable sections: 'Directory Settings' and 'Special Attributes'. At the bottom of the form, there are three buttons: 'Save', 'Test Configuration', and 'Help'.

Figure 3.1: 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. This 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 off 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 exist 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 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.

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

- User Display Name Field: `displayname`
- Base User Tree: `dc=owncloud,dc=bzoc`
- User Search Attributes: `Optional; one attribute per line`
- Group Display Name Field: `cn`
- Base Group Tree: `dc=owncloud,dc=bzoc`
- Group Search Attributes: `Optional; one attribute per line`
- Group-Member association: `uniqueMember`

At the bottom of the form are three buttons: 'Save', 'Test Configuration', and 'Help'.

Figure 3.2: 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

The screenshot shows the 'Advanced' tab of the LDAP configuration interface. Under the 'Special Attributes' section, there are four input fields: 'Quota Field', 'Quota Default', 'Email Field', and 'User Home Folder Naming Rule'. Below these fields are three buttons: 'Save', 'Test Configuration', and 'Help'.

Figure 3.3: 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

The screenshot shows the 'Expert' settings tab in the ownCloud configuration wizard. It includes the following sections:

- Internal Username:** A text box for 'Internal Username Attribute' with explanatory text about UUID-based usernames.
- Override UUID detection:** Two text boxes for 'UUID Attribute for Users' and 'UUID Attribute for Groups' with explanatory text about overriding the default UUID detection.
- Username-LDAP User Mapping:** Two buttons: 'Clear Username-LDAP User Mapping' and 'Clear Groupname-LDAP Group Mapping'.

At the bottom of the form are three buttons: 'Save', 'Test Configuration', and 'Help'.

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. 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

The **Test Configuration** button on the bottom of the LDAP settings section 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 3.4: 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.

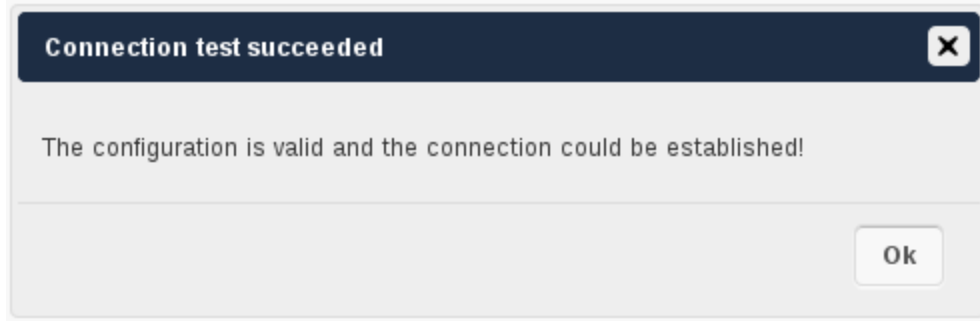


Figure 3.5: Success

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*.

#### Profile picture



Your avatar is provided by your original account.

Figure 3.6: 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 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. The cache stores 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, it 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. For planned downtime, check **Disable Main Server** for the time being to avoid unnecessary connection attempts every now and then.

## 3.4 Background Jobs

A system like ownCloud sometimes requires tasks to be done on a regular base without blocking the user interface. For that purpose you, as a system administrator, can define background jobs which make it possible to execute tasks without any need of user interaction, e.g. database clean-ups etc. For the sake of completeness it is worth to know that additionally background jobs can also be defined by installed apps.

### 3.4.1 Parameters

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

- AJAX
- Webcron
- Cron

### 3.4.2 Cron-Jobs

OwnCloud requires various automated background jobs to be run. There are three methods to achieve this. The default way is AJAX and the recommended way is cron.

## AJAX

This option is the default option, although it is the least reliable. Every time a user visits the ownCloud page a single background job will be executed. The advantage of this mechanism is, that it does not require access to the system nor registration at a third party service. The disadvantage of this solution compared to the Webcron service is, that it requires regular visits of the page to get triggered.

## Webcron

By registering your ownCloud `cron.php` script address at an external webcron service, like e.g. [easyCron](#), you ensure that background jobs will be executed regularly. To use such a service your server needs to be reachable via the Internet.

### Example

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

## Cron

Using the systems cron feature is the preferred way to run regular tasks, because it allows to execute jobs without the limitations which a web server may have.

### Example

To run a cron job on a \*nix system, e.g. every 15min, under the default web server user, e.g. **www-data**, you need to set-up the following cron job to call the **cron.php** script. Please check the crontab man page for the exact command syntax.

```
# crontab -u www-data -e
*/15 * * * * php -f /var/www/owncloud/cron.php
```

## 3.5 Find Third-Party Libraries

ownCloud resorts to some 3rd-party PHP components to provide its functionality. These components are part of the software package and are usually shipped in the **/3rdparty** folder.

### 3.5.1 Parameters

If you want to change the default location of the 3rd-party folder you can use the **3rdpartyroot** parameter to define the absolute file system path to the folder. The **3rdpartyurl** parameter is used to define the http web path to that folder, starting at the ownCloud web root.

```
<?php
```

```
"3rdpartyroot" => OC::$SERVERROOT."/3rdparty",
"3rdpartyurl"  => "/3rdparty",
```

## 3.6 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 [MySQL/Postgres/SQLite Support](#). For this reason the automatic configuration feature has been introduced.

To take advantage of this feature you need to create a configuration file, called `../owncloud/config/autoconfig.php` and set the parameters as required. You can provide all parameters or just part of them - parameters which haven't been provided (if any) will be asked at "Finish setup" screen at first run of ownCloud.

The `../owncloud/config/autoconfig.php` will be automatically removed after the initial configuration has been applied.

### 3.6.1 Parameters

You need to keep in mind that two parameters are named differently in this configuration file compared to the normal `config.php`.

<code>autoconfig.php</code>	<code>config.php</code>
<code>directory</code>	<code>datadirectory</code>
<code>dbpass</code>	<code>dbpassword</code>

### 3.6.2 Sample Automatic Configurations

#### Data Directory

With the configuration below the "Finish setup" screen still will ask for database and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
    "directory" => "/www/htdocs/owncloud/data",
);
```

#### SQLite Database

With the configuration below the "Finish setup" screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
    "dbtype" => "sqlite",
    "dbname" => "owncloud",
    "dbtableprefix" => "",
);
```

#### MySQL Database

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in [MySQL/Postgres/SQLite Support](#).

With the configuration below the "Finish setup" screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
    "dbtype"      => "mysql",
    "dbname"      => "owncloud",
    "dbuser"      => "username",
    "dbpass"      => "password",
    "dbhost"      => "localhost",
    "dbtableprefix" => "",
);
```

## PostgreSQL Database

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in *MySQL/Postgres/SQLite Support*.

With the configuration below the “Finish setup” screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
    "dbtype"      => "pgsql",
    "dbname"      => "owncloud",
    "dbuser"      => "username",
    "dbpass"      => "password",
    "dbhost"      => "localhost",
    "dbtableprefix" => "",
);
```

## All Parameters

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in *MySQL/Postgres/SQLite Support*.

With the configuration below “Finish setup” will be skipped at first ownCloud run since all parameters are already preconfigured.

```
<?php
$AUTOCONFIG = array(
    "dbtype"      => "mysql",
    "dbname"      => "owncloud",
    "dbuser"      => "username",
    "dbpass"      => "password",
    "dbhost"      => "localhost",
    "dbtableprefix" => "",
    "adminlogin"  => "root",
    "adminpass"   => "root-password",
    "directory"   => "/www/htdocs/owncloud/data",
);
```

## 3.7 Customizing Client Download Links

If you want to access your ownCloud, you can choose between the standard Web-GUI and different client sync applications. Download links which point to these applications are shown at the top of the personal menu. The following sync applications are currently available out of the box:

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

### 3.7.1 Parameters

If you want to customize the download links for the sync clients the following parameters need to be modified to fulfil your requirements:

```
<?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",
```

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

## 3.8 MySQL/Postgres/SQLite Support

ownCloud requires a database where administrative data will be held. Four different database types are currently supported, [MySQL](#), [MariaDB](#), [SQLite](#), and [PostgreSQL](#). MySQL or MariaDB are the recommended database engines. By default SQLite is chosen because it is a file based database with the least administrative overhead.

---

**Note:** Because SQLite handles multiple users very badly SQLite is only recommended for single user ownCloud installations

---

### 3.8.1 Requirements

If you decide to use MySQL, MariaDB, or PostgreSQL you need to install and set-up the database first. These steps will not be covered by this description as they are easy to find elsewhere.

### 3.8.2 Parameters

#### MySQL/MariaDB Database

If you decide to use a MySQL or MariaDB database make sure that you have installed and enabled the MySQL extension in PHP and that the `mysql.default_socket` points to the correct socket (if the database runs on same server as ownCloud).

Please note that MariaDB is backwards compatible with MySQL, so all instructions will 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
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username and password to access it.

```
<?php
```

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

## 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.

In the ownCloud configuration in `config/config.php` you need to set at least the **datadirectory** parameter to the directory where your data and database should be stored. Note that for the PDO SQLite driver this directory must be writable (this is recommended for ownCloud anyway). No authentication is required to access the database therefore most of the default parameters could be taken as is:

```
<?php

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

## PostgreSQL Database

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

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

### [PostgreSQL]

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

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
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username (and sometimes a password) to access it. If the database has been installed on the same server as ownCloud a password is very often not required to access the database.

```
<?php

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

## 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
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username and password to access it. If the database has been installed on the same server as ownCloud to config file could look 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`.

---

### 3.8.3 Trouble Shooting

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

Use the ping command to check the server availability:

```
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
```

#### 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:

```
mysql -uUSERNAME -p
```

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

##### PostgreSQL:

```
psql -Uusername -downcloud
```

```
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:

```
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
```

## 3.9 Using Server-Side Encryption

ownCloud ships a encryption app, which allows to encrypt all files stored in your ownCloud. Encryption and decryption always happens server-side. This enables the user to continue to use all the other apps to view and edit his data.

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)

### 3.9.1 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 enables 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.

### 3.9.2 Recover User Files

If the recovery feature was enabled the admin will see a 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.

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

if you configure a 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.

## 3.10 Disable Knowledge Base

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.

### 3.10.1 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

---

## 3.11 Setting the Default Language

In normal cases ownCloud will automatically detect the language of the Web-GUI. If this doesn't 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.

### 3.11.1 Parameters

```
<?php
```

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

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

## 3.12 Configure Logging

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.

### 3.12.1 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

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",
```

## **3.13 Sending Mail Notifications**

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.

### **3.13.1 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**.

### **3.13.2 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" => "",
```

### 3.13.3 Send a Test Email

The only way to test your email configuration is, to force a login failure, because a function to send a test email has not been implemented yet.

First make sure that you are using a full qualified domain and not an ip address in the ownCloud URL, like:

```
http://my-owncloud-server.domain.dom/owncloud/
```

The password reset function fetches the domain name from that URL to build the email sender address, e.g.:

```
john@domain.dom
```

Next you need to enter your login and an *invalid* password. As soon as you press the login button the login mask reappears and a **I've forgotten my password** link will be shown above the login field. Click on that link, re-enter your login and press the **Reset password** button - that's all.

### 3.13.4 Trouble shooting

#### My web domain is different from my mail domain?

The default domain name used for the sender address is the hostname where your ownCloud installation is served. If you have a different mail domain name you can override this behavior by setting the following configuration parameter:

```
<?php
```

```
"mail_domain" => "example.com",
```

Now every mail sent by ownCloud e.g. password reset email, will have the domain part of the sender address look like:

```
no-reply@example.com
```

#### How can I find out if a SMTP server is reachable?

Use the ping command to check the server availability:

```
ping smtp.server.dom
```

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

#### How can I find out if the SMTP server is listening on a specific tcp port?

A SMTP server is usually listening on port **25/tcp** (smtp) and/or in rare circumstances is also listening on the outdated port **465/tcp** (smtps). You can use the telnet command to check 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
```

### How can I find out if a SMTP server supports the outdated SMTPS protocol?

A good indication that a SMTP server supports the SMTPS protocol is that it is listening on port **465/tcp**. How this can be checked has been described previously.

### How can I find out if a SMTP server supports the TLS protocol?

A SMTP server usually announces the availability of STARTTLS right after a connection has been established. This can easily be checked with the telnet command. You need to enter the marked lines to get 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 # <<< enter this command
QUIT # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

### How can I find out which authentication types/methods a SMTP server supports?

A SMTP server usually announces the available authentication types/methods right after a connection has been established. This can easily be checked with the telnet command. You need to enter the marked lines to get 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 # <<< enter this command
QUIT # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

## Enable Debug Mode

If you are still not able to send email it might be useful to activate further debug messages by setting the following parameter. Right after you have pressed the **Reset password** button, as described before, a lot of **SMTP -> get\_lines():** ... messages will be written on the screen.

```
<?php
"mail_smtpdebug" => true;
```

## 3.14 Enable Maintenance Mode

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.

### 3.14.1 Parameters

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

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

## 3.15 Enabling File Previews

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

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

## **3.15.2 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' => ' ',
```

## 3.16 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.

### 3.16.1 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.

### 3.16.2 Example

#### Multiple Domains Reverse SSL Proxy

If you want to access your ownCloud installation **http://domain.tld/owncloud** via a multiple domains reverse SSL proxy **https://ssl-proxy.tld/domain.tld/owncloud** with the IP address **10.0.0.1** you can set the following parameters inside the `config/config.php`.

```
<?php
$CONFIG = array (
    "overwritehost"      => "ssl-proxy.tld",
    "overwriteprotocol" => "https",
    "overwritewebroot"  => "/domain.tld/owncloud",
```

```
"overwritecondaddr" => "^10\.0\.0\.1$",  
);
```

---

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

---

## 3.17 Dealing with Big File Uploads

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

### 3.17.1 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

### 3.17.2 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.

### 3.17.3 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-WebUI and do the following:**

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

**Open the `php.ini` - file**

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

**Do the following:**

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



whereas 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

## 3.18 Custom Mount Configuration Web-GUI

Since ownCloud 5.0 it is possible to mount external storage providers into ownCloud’s virtual file system. To add an external storage backend to your ownCloud head to *Settings -> Admin* or *Personal*. As administrator you can mount external storage for any group or user. Users are also allowed to mount external storage for themselves if this setting has been enabled by the administrator.

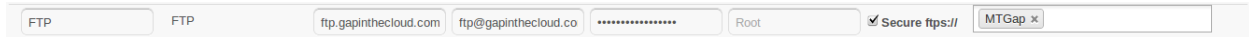
At first the mount point has to be entered, this is the directory in ownCloud’s virtual file system, that the storage will be mounted to. Then the storage backend has to be selected from the list of supported backends. As of writing ownCloud currently supports the following storage backends:

- Local file system (mount local storage that is outside ownCloud’s data directory)
- FTP (or FTPS)
- SFTP
- SMB
- WebDAV
- Amazon S3
- Dropbox
- Google Drive
- OpenStack Swift

Please keep in mind, that users are not allowed to mount local file storage for security purposes.

Once a backend has been selected, more configuration fields will appear. The displayed configuration fields may vary depending on the selected storage backend. For example, the FTP storage backend needs the following configuration details to be entered:

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



- **user:** the username used to login to the ftp server
- **password:** the password to login to the ftp server
- **secure:** whether to use ftps:// (FTP over TLS) to connect to the ftp server instead of ftp:// (optional, defaults to false)
- **root:** the name of the folder inside the ftp server to mount (optional, defaults to '/')

### 3.18.1 Dropbox

Mounting a Dropbox account requires that you create an app with Dropbox and then provide the app key and secret to the external storage configuration user interface. Go to My apps at Dropbox and create an app. Select *Full Dropbox* access level. Copy the app key and app secret and paste them into the corresponding fields for the Dropbox storage.

Click the *Grant access* button and you will be redirected to a Dropbox website to give ownCloud permission to access your account.

### 3.18.2 Google Drive

For a detailed step-by-step guide read User Manual

## 3.19 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!

### 3.19.1 Example

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

### 3.19.2 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" }
}
```

---

**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"
  }
}
```

---

**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"
  }
}
```

---

**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"
  }
}
```

## 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"
  }
}
```

## 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"
  }
}
```

## 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"
  }
}
```

## 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"
  }
}
```

#### 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"
  }
}
```

## 3.20 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:

### 3.20.1 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}'
        ),
    ),
),
```

### 3.20.2 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'
        ),
    ),
),
```

## 3.21 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. Controlled file serving **does not work for generated zip files**. This is due to how temporary files are created.

### 3.21.1 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 /tmp/oc-noclean
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 PHP temporary directory where zip files are created and 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. At least PHP temporary directory concatenated with *oc-noclean* must be configured. Temporary zip files will be created inside this directory when using `mod_xsendfile`. Other paths which correspond to local mounts should be configured here as well. For a more in-dept documentation of this directive refer to `mod_xsendfile` website linked above

### 3.21.2 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

### 3.21.3 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-Accel](#). 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 ~ ^/home/valerio/(owncloud/)?data {
    internal;
    root /;
}

location ~ ^/tmp/oc-noclean/.+$ {
    internal;
    root /;
}
```

- **fastcgi\_param MOD\_X\_ACCEL\_REDIRECT\_ENABLED:** tells ownCloud scripts that they should add the X-Accel-Redirect header when serving files
- **internal location:** each directory that contains local user data should correspond to an internal location. In the example uses the following directories:
  - **/home/valerio/owncloud/data:** ownCloud data directory
  - **/home/valerio/data:** a local mount
  - **/tmp/oc-noclean:** PHP temporary directory concatenated with *oc-noclean*. Temporary zip files will be created inside this directory when using X-Accel-Redirect

### 3.21.4 How to check if it's working?

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



## 4.1 Activity

The ownCloud Activity app shows an activity feed where users can see what they did in the past and what happened to their files and what others did with their shared files.

The Activity Stream introduces a central interface where all events that happen in a user's ownCloud instance are shown.

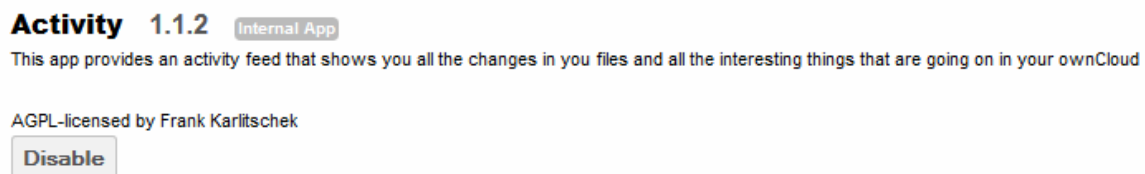
The following can be seen in the new interface:

- Creation, edit or deletion of files along with a thumbnail and date and path
- If someone shared a file with you
- If someone created, edited or deleted something in a folder you shared to them.

All the events are presented with endless scrolling and also in an RSS feed which a user can subscribe to.

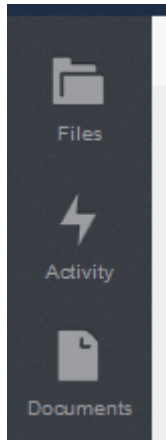
### 4.1.1 Configuration

The Activity App is enabled by default. To verify or to disable, navigate to the APPS page and search for `Activity`.



### 4.1.2 Utilization

The Activities can be viewed by selecting Activity in the App bar on the left side of the ownCloud browser interface



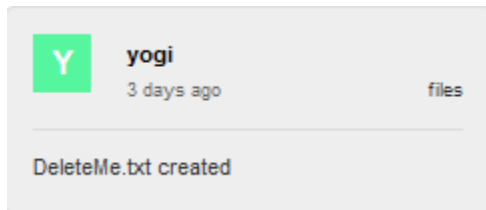
This will bring up the Activity interface showing all the activities which occurred for this user's instance.

## Sample Activities

This section will show how the different activities may appear within this interface

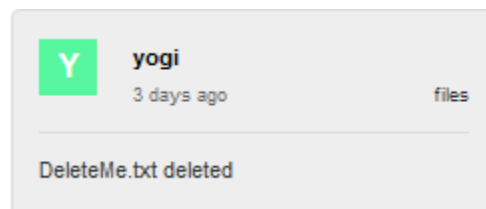
### Create file

The following shows a file entitled DeleteMe.txt being created 3 days ago.



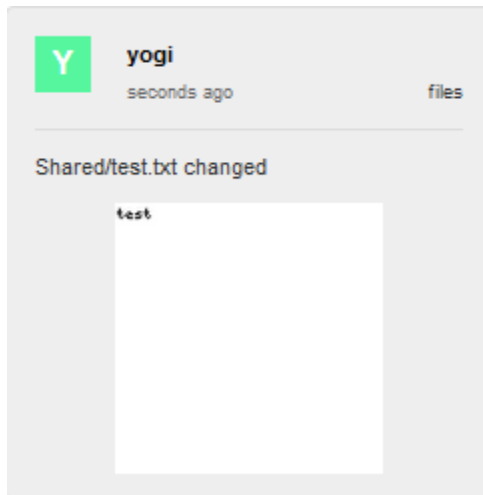
### Delete File

The following shows a file entitled DeleteMe.txt being deleted 3 days ago.



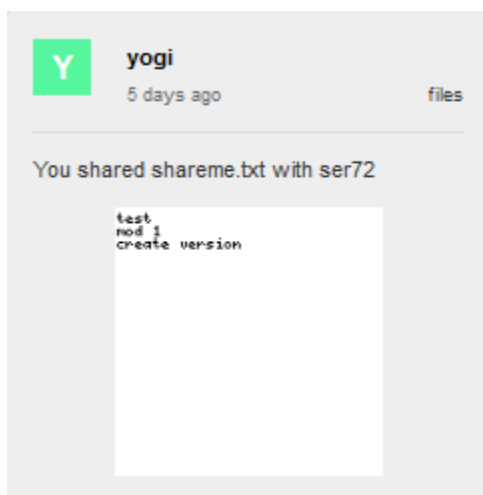
### Edit a file

The following shows a file Shared/test.txt being edited mere seconds ago.



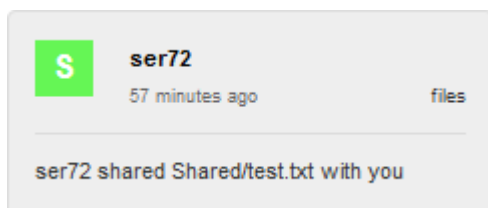
### Share a file with a user

The following shows the file `shareMe` being shared with another user.



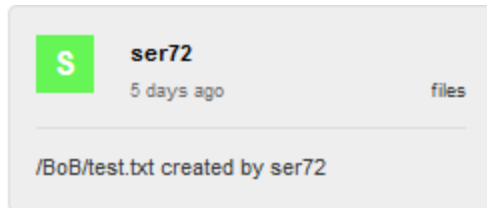
### File shared to user

The following shows the file `Shared/test.txt` being shared with this user almost an hour ago.



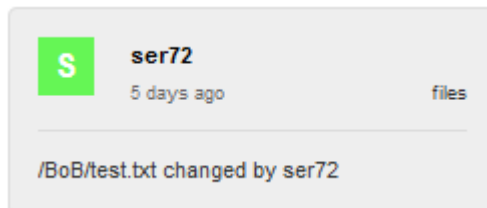
### File created by another user in Shared folder

The following is shown when another user creates a file in your shared folder.



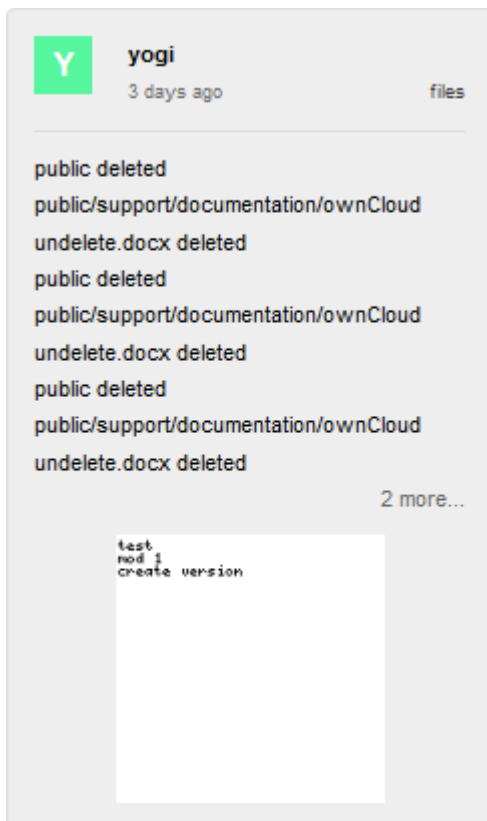
### Shared file modified

The following shows a file `BoB/test.txt` modified by a user who this file was shared to.



### Combined activities

The following shows what “combined” activities may look like on the activity interface.



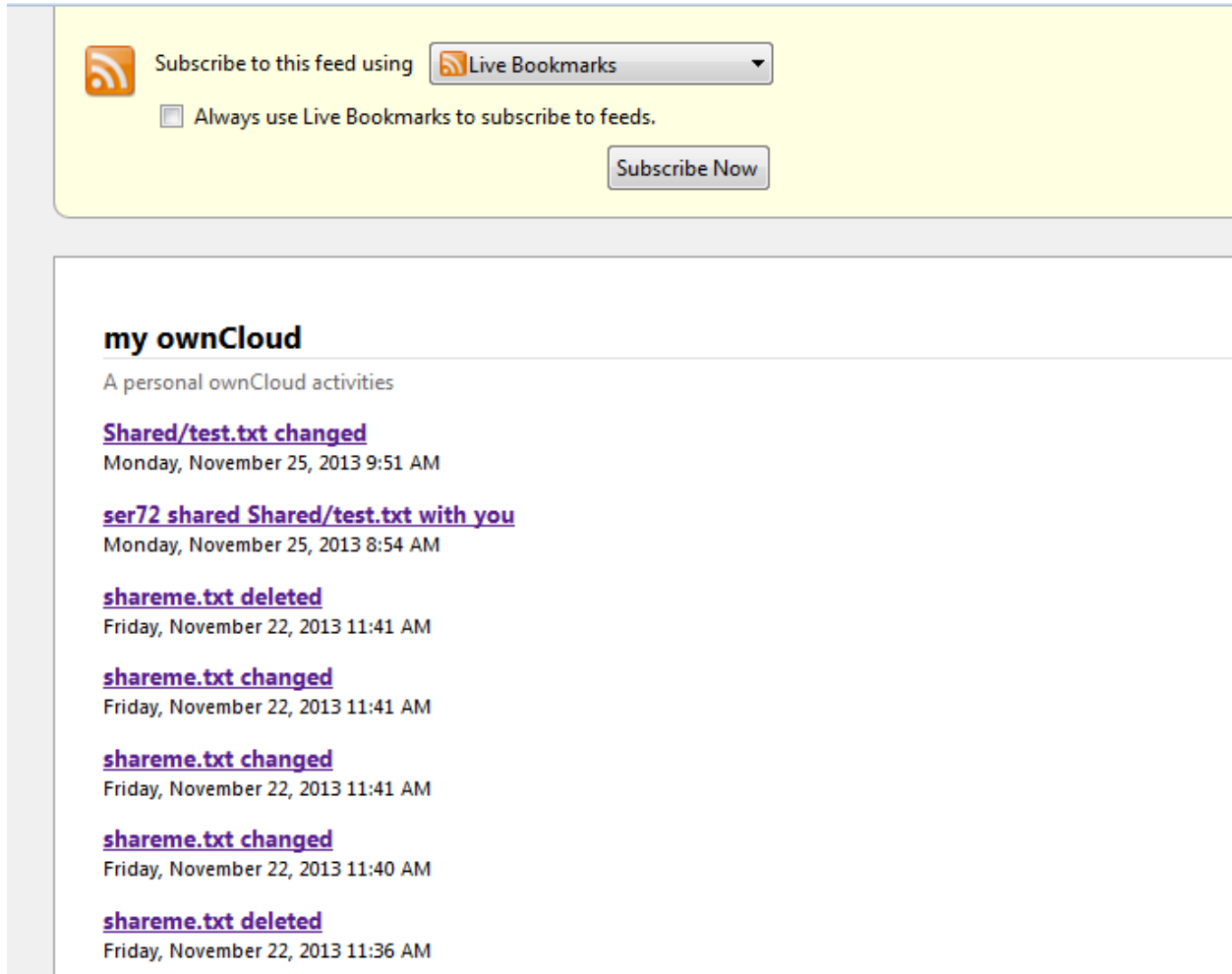


### 4.1.3 RSS Feed

To subscribe to the RSS Feed, navigate to the `Activity` page and select `RSS feed` in the upper right of the browser.



The resulting page will look something like this:



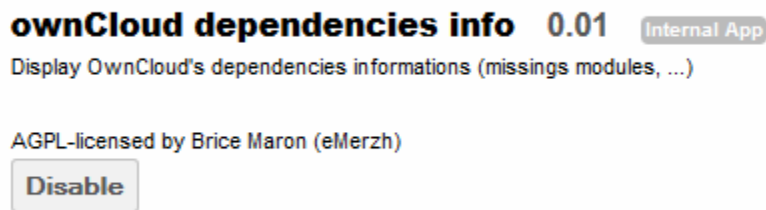
To subscribe, select the `Subscribe Now` button. The RSS Feed may be viewed via the browser's capabilities.

## 4.2 Finding Deployment Dependencies

The ownCloud Dependencies App provides a list of modules required to run the current setup of ownCloud. All of the modules listed should be installed on the base operating system prior to using ownCloud.

### 4.2.1 Configuration

By default, the ownCloud Dependencies App is disabled. To enable, navigate to the Apps page and select “ownCloud dependencies info” and enable.



### 4.2.2 Utilization

Once this app is enabled, navigate to the Admin page and scroll to “Dependencies status”. This section will show a list of all modules required to execute this ownCloud setup as well as what part of ownCloud uses the given module.

- **php-json**  
Used by : core
- **php-gd**  
Used by : gallery
- **php-zip**  
Used by : admin\_export core
- **php-mb\_multibyte**  
Used by : core
- **php-ctype**  
Used by : core
- **php-xml**  
Used by : core
- **allow\_url\_fopen**  
Used by : core
- **php-pdo**  
Used by : core
- **php-iconv**  
Used by : files\_texteditor news contacts

Modules in green are required and have been installed. Modules in red are required yet have not been installed. It is recommended to install these modules prior to using ownCloud.

## 4.3 File Antivirus Engine

ownCloud integrates with ClamAV, an open source (GPL) antivirus engine, to provide an antivirus solution for files which are uploaded to the ownCloud server. Via this method, ownCloud/ClamAV can detect Trojans, viruses, malware and other malicious threats. Files are scanned for virus upon initial upload to the ownCloud server.

The ownCloud antivirus app is supported on ownCloud instances which are installed on a Linux operating system.

The antivirus app can run in one of three modes:

- Executable – ClamAV is running on the same server as the ownCloud instance. For executable mode, the ClamAV process is started and stopped with each file upload.
- Daemon – ClamAV is running on a different server from the ownCloud instance
- Daemon (Socket) – ClamAV is running on the same server as the ownCloud instance. In this mode, the ClamAV process is running in the background at all times. It is a bit quicker for scanning than executable mode, but requires system administrator skills and root access.

In addition, there are two possible actions which may occur when an infected file is found:

- Only Log – A log entry is created when an infected file is found.
- Delete File – The infected file is deleted.

### 4.3.1 Configuration

#### Enable The app

To enable the Antivirus App, navigate to the Apps page and select “Antivirus App for Files” and enable.

#### **Antivirus App for files 0.4**

Verify files for virus using ClamAV

[See application page at apps.owncloud.com](https://apps.owncloud.com)

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

Enable

#### Install ClamAV

ClamAV must be installed on the server (either the local for Executable or Daemon Socket mode or a remote server for Daemon mode).

To install, use the repository’s installation method to install “clamav”. For example:

```
apt-get install clamav
```

For daemon mode, the ClamAV daemon must also be installed (either on the local machine for Daemon Socket or the remote machine for daemon mode):

```
apt-get install clamav-daemon
```

## Configure Logging

Set log level to Everything in the Admin page.

### Log

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

## Executable Mode

To run in executable mode, ClamAV must be installed on the local server. From the Admin page, configure Antivirus as follows:

### Antivirus Configuration

Mode **Executable** ▼

Stream Length  bytes

Path to clamscan

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

**Save**

The Stream Length is defined as the ClamAV StreamMaxLengeth Size. The default value, according to the ClamAV web site is 10M which equates to 10485760 bytes as shown in the above example.

The Path to clamscan is the path for the executable clamscan file. By default it installs in /usr/bin/clamscan.

When files are uploaded, they will be scanned and, if clean, the following logs will appear:

```
{"app":"files_antivirus","message":"Scanning file : \\Lab.txt","level":0,"time":"2013-12-17T15:24:05-05:00"}
{"app":"files_antivirus","message":"Exec scan: \\Lab.txt","level":0,"time":"2013-12-17T15:24:05+00:00"}
{"app":"files_antivirus","message":"Result CLEAN!","level":0,"time":"2013-12-17T15:24:09+00:00"}
```

## Daemon Mode

When running in Daemon Mode, install ClamAV and clamAV-Daemon on a remote server.

The port, upon which ClamAV listens must be configured. To do this, add the following line in /etc/clamav/clamd.conf:

```
TCPsocket 3310
```

Then restart the Clamd service:

```
/etc/init.d/clamav-daemon restart
```

Back on the ownCloud server, navigate to the Admin page and configure the Antivirus Configuration as follows:

## Antivirus Configuration

Mode **Daemon** ▼

Host

Port

Stream Length  bytes

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

**Save**

Where the host is the IP of the server running the ClamAV Daemon and the Port is what was configured in the above step.

Upon upload of files to the ownCloud server, the following logs will appear indicating the files are clean:

```
{ "app": "files_antivirus", "message": "Scanning file : \\Lab.txt", "level": 0, "time": "2013-12-17T17:39:35-0800" }
{ "app": "files_antivirus", "message": "Response :: stream: OK\\n", "level": 0, "time": "2013-12-17T17:39:48-0800" }
```

### Daemon Socket mode

To run in Daemon socket mode, install clamav and clamav-daemon on the ownCloud server.

Configure the Admin page as such:

## Antivirus Configuration

Mode **Daemon (Socket)** ▼

Socket

Stream Length  bytes

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

**Save**

Where Socket is the location of the Clamd executable.

Upon upload of a clean file to the ownCloud server, the following logs will appear:

```
{ "app": "files_antivirus", "message": "Scanning file : \\Lab.txt", "level": 0, "time": "2013-12-17T18:19:08-0800" }
{ "app": "files_antivirus", "message": "Response :: stream: OK\\n", "level": 0, "time": "2013-12-17T18:19:08-0800" }
```

## 4.4 Encryption

ownCloud contains an encryption app which, when enabled, encrypts all files stored in ownCloud. The encryption is done automatically once the admin enables the app. All encryption and decryption occur on the ownCloud server, which allows the user to continue to use other apps to view and edit the data.

The user's password is used as the key to decrypt their data. This means that if the user loses their login password, data will be lost. To protect against password loss, the recovery key may be used as described in a later section.

### 4.4.1 What gets encrypted?

All files stored in ownCloud will be encrypted with the following exceptions:

- Old versions (versions created prior to enabling the encryption app)
- Old files in the trash bin (files deleted prior to enabling the encryption app)
- Existing files on external storage. Only new files placed on the external storage mount after encryption was enabled are encrypted.
- Image thumbnails from the gallery app
- Search index from the full text search app.

### 4.4.2 Decrypting the data

If the encryption app is disabled, users will get the following message alerting them how to decrypt their files.

Encryption was disabled but your files are still encrypted. Please go to your personal settings to decrypt your files.

Navigating to the Personal settings page, the user can enter their password and decrypt all files.

#### Encryption

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

Log-in password

Decrypt all Files

### 4.4.3 Configuration

To enable the encryption app, navigate to the Apps page and select Encryption, then enable.

## Encryption 0.5 Internal App

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

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

**Enable**

Once the app is enabled, the following message will appear for all users currently logged into the web browser as they navigate to a new page within the ownCloud web browser

**Encryption App is enabled but your keys are not initialized, please log-out and log-in again**

It is necessary to logout of ownCloud and re-login to initialize the encryption keys.

### Recovery Key

Enabling the recovery key globally is done by the admin. Each user then has the option as to whether they wish to do so for their own account. If enabled, the admin will be able to reset the user's encryption password using a predefined recovery password. This allows for the recovery of a user's files in the event of a password loss. If recovery key is not enabled, there is no way to restore files if the login password is lost.

#### Admin level

To enable the recovery key, the Admin must first enable this feature in the Admin page.

## Encryption

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

Recovery key password

Repeat Recovery key password

☐ Enabled  
☒ Disabled

Once enabled, the Admin may change the recovery key password at any time.

## Encryption

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

 Recovery key password  
 Repeat Recovery key password  

☒ Enabled

☐ Disabled

Change recovery key password:

 Old Recovery key password  
 New Recovery key password  
 Repeat New Recovery key password  

Change Password

### User level

The user can then navigate to the Personal page and enable password recovery.

## Encryption

Enable password recovery:

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

- ☒ Enabled
- ☐ Disabled



## Recovery

In order for the admin to recover the user's files in the event of a lost password, the admin should navigate to the "Users" tab and enter the Recovery Key Password into the "Admin Recovery Password" field at the top of the page.

Login Name	Password	Groups ▼	Create	Admin Recovery Password	Default Storage	Unlimited ▼
------------	----------	----------	--------	-------------------------	-----------------	-------------

This will allow the admin to generate a new encryption password for the user.

- For local user management, this will generate both a new log-in password as well as a new encryption password for the data.
- If LDAP authentication is in use, the admin would need to set the new log-in password in the LDAP server, and then enter the same password in the user management page for the encryption key. It is important that both the password in the LDAP server and the password entered in the user management page are identical.

### 4.4.4 File Systems

Once enabled, all files within ownCloud are encrypted, with the exceptions mentioned above. This includes files in local storage, as well as files contained within external storage mounts.

The encryption app creates several key files/folders when enabled. `~/data/public-keys` contains the public keys for all users, and `~/data/owncloud_private_keys` contains system wide private keys utilized for public link shares as well as the recovery key.

```
root@server:/var/www/owncloud/data# ls
files_encryption/  mount.json  owncloud.log          public-keys/  yogi/
index.html        oc6admin/   owncloud_private_key/  user1/
```

The encryption app stores key information in the `~/data/<user>/files_encryption` directory.

```
root@server:/var/www/owncloud/data/user1/encryption# ls
keyfiles/  user1.private.key  share-keys/
```

As mentioned previously, the private key is generated from the user's password.

Each file that the user owns will have a corresponding keyfile maintained in the keyfiles directory.

```
root@server:/var/www/owncloud/data/user1/files_encryption/keyfiles# ls
documents/  ownCloud  undelete.docx.key  photos/
music/      ownCloudUserManual.pdf.key  test  encryption.txt.key
```

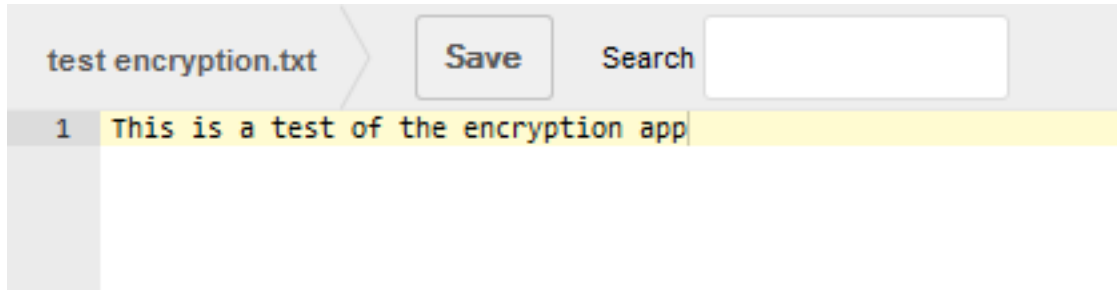
In addition a share key will be generated for each file in the event that there is an external storage mount by the admin for multiple users or groups.

```
root@server:/var/www/owncloud/data/user1/files_encryption/share-keys# ls
documents/
music/
ownCloud  undelete.docx.recovery_5dcce10a.shareKey
ownCloud  undelete.docx.user1.shareKey
ownCloudUserManual.pdf.recovery_5dcce10a.shareKey
ownCloudUserManual.pdf.user1.shareKey
photos/
test  encryption.txt.recovery_5dcce10a.shareKey
test  encryption.txt.user1.shareKey
...
```

When viewing a file directly on the ownCloud data directory, it will show up as encrypted.

```
root@server: /var/www/owncloud/data/user1/files# more test\ encryption.txt
2JnmDdDh//8FVcDhLrnDlWH0JjhrzKpFKV6V61pAfUCu9IJX00iv007Yw3Tf/QBbtJFpQFxx
```

However, viewing the same file via the browser, the actual contents of the file are displayed.



## 4.5 External storage support

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.

### 4.5.1 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

### 4.5.2 Configuration

#### Enable the app

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

## External storage support 0.2 Internal App

Mount external storage sources

AGPL-licensed by Robin Appelman, Michael Gapczynski

**Enable**

### 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

### External Storage

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

## Amazon S3

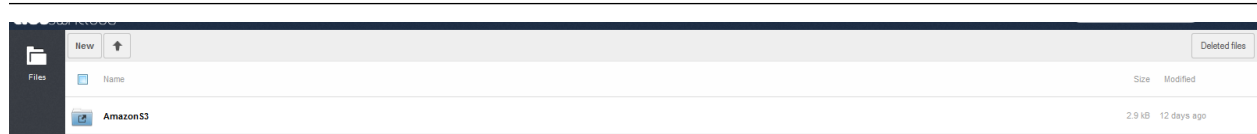
This is used to mount to an S3 server

External Storage		Configuration		Applicable						
Folder name	External storage	Access Key	Secret Key	Bucket	Hostname (optional)	Port (optional)	Region (optional)	Enable SSL	Enable Path Style	Applicable
AmazonS3	Amazon S3							<input type="checkbox"/>	<input type="checkbox"/>	None set

- 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

External Storage		Configuration		Applicable						
Folder name	External storage	Access Key	Secret Key	Bucket	Hostname (optional)	Port (optional)	Region (optional)	Enable SSL	Enable Path Style	Applicable
 AmazonS3	Amazon S3	KIAJILSTB6YSDN7XQIQ	*****	ocsteve				<input type="checkbox"/>	<input type="checkbox"/>	ser72, x

**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:





Developer home

App Console

This will ask you to accept terms and conditions.

Select Dropbox API and configure down the page as follows:

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

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

☒ Files and datastores  
☐ Datastores only

Can your app be limited to its own, private folder?

☐ Yes — My app only needs access to files it creates.  
☒ No — My app needs access to files already on Dropbox.

What type of files does your app need access to?

☐ Specific file types — My app only needs access to certain file types, like text or photos.  
☒ All file types — My app needs access to a user's full Dropbox. Only supported via the [Core API](#).

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

This app name is already taken

The name can be any unique name desired.

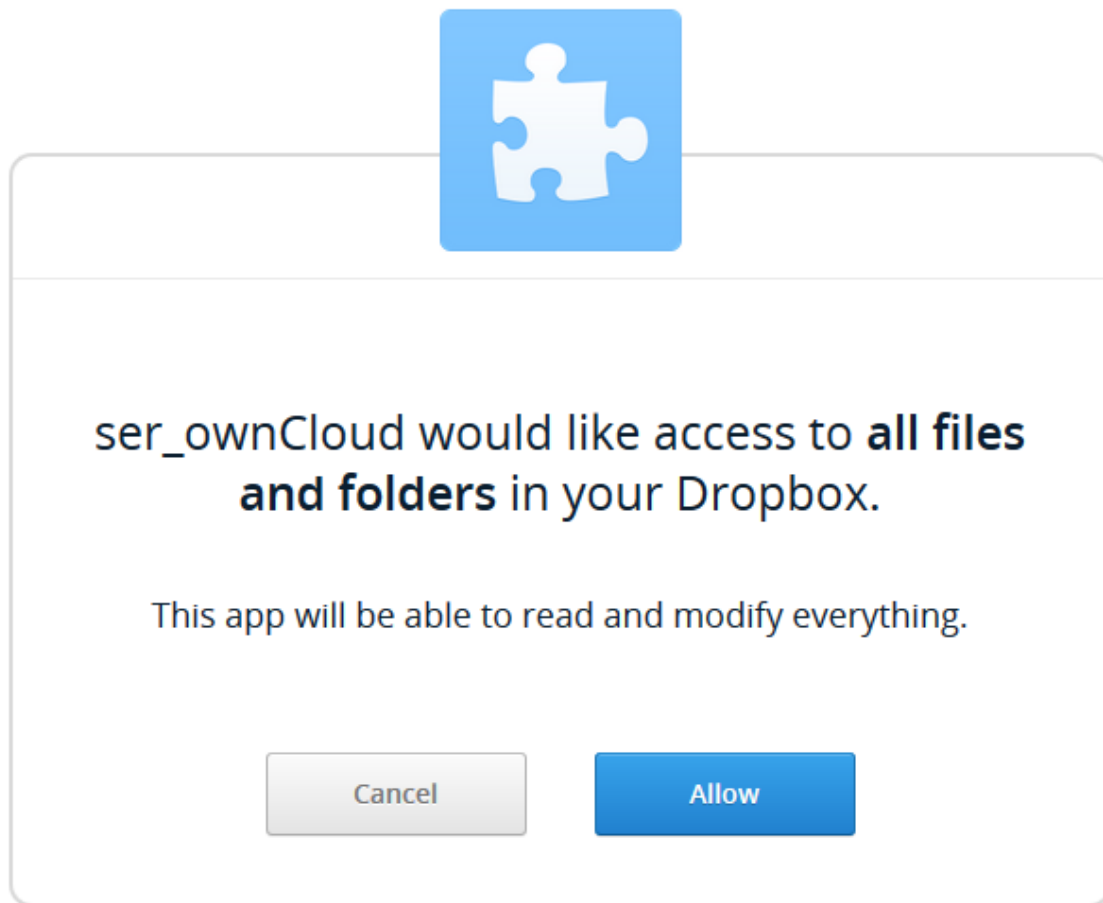
Select Create App

Create app

Enter the OAuth redirect URI as follows:

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





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

External Storage					Applicable
Folder name	External storage	Configuration			
 Dropbox	Dropbox	vyhe5udkxien2ps	ciqlyvi1wrha4sv	Access granted	<a href="#">All Users</a> x

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

 Name	Size	Modified
 Dropbox	?	seconds ago


## FTP

Mounts a folder on a remote FTP or FTPS server




External Storage					Applicable
Folder name	External storage	Configuration			
FTP	FTP	URL	Username	Password	Root
				<input checked="" type="checkbox"/> Secure ftps://	None set

- 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 server
- 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

External Storage							
Folder name	External storage	Configuration			Applicable		
 FTP	FTP	192.168.1.68	anonymous	*****	/	<input checked="" type="checkbox"/> Secure ftps://	<input type="text" value="ser72 x"/>

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

 Name	Size	Modified
 documents	22.8 kB	2 hours ago
 FTP	0 B	years ago

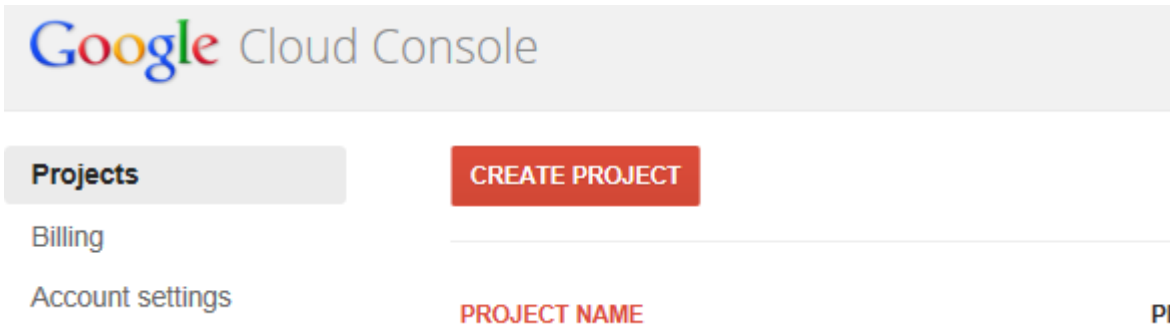
## GoogleDrive

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

New Project

Project name ?

ownCloud

Project ID ?

enhanced-kiln-418

Create

Cancel

ownCloud

Overview

APIs & auth

APIs

Registered apps

Consent screen

Notification endpoints

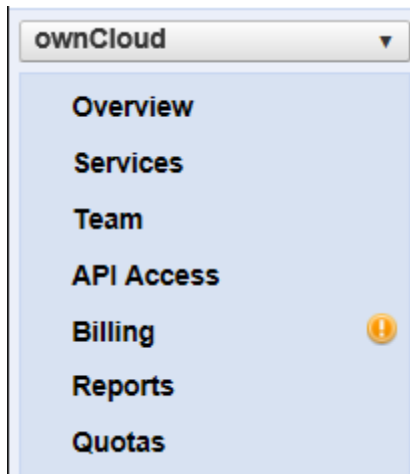
Select the project and choose the APIs & auth menu entry  
Enable Drive API and Drive SDK and then select the



next to either Drive API or Drive SDK

Drive API		ON
Drive SDK		ON

Select API Access on the menu



Select REGISTER APP

< ownCloud

REGISTER APP

Overview

NAME

APIs & auth

Service Account-project

APIs

Registered apps

Consent screen

Enter a name and select Web Application

## Register new application

You need to register your application to get the necessary credentials to call a Google API.

Name	<input type="text" value="owncloud"/>
Platform	<input checked="" type="radio"/> Web Application <input type="radio"/> Android <input type="radio"/> iOS <input type="radio"/> Chrome <input type="radio"/> Native Windows Mobile, Blackberry, desktop, devices, and more
<input type="button" value="Register"/>	

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

▼ OAuth 2.0 Client ID

Access user data via a consent screen

Download JSON

CLIENT ID

858877404875-n7juj6b6go5ea1b4fmsenne0p7mgi3m7.apps.googleusercontent.com

CLIENT SECRET

nNOjXtqTjg\_irvZpRYXI2Te

CONSENT SCREEN

Update

WEB ORIGIN

REDIRECT URI

<http://s3fs.ser.net/s3store/index.php/settings/admin> - +  
<http://s3fs.ser.net/s3store/index.php/settings/personal> - +

Generate

Verify that the required email addresses are in the Permissions tab

< ownCloud

ADD MEMBER

Overview

APIs & auth





Permissions

Settings

Support

App Engine Preview

Compute Engine

EMAIL	PERMISSION
 858877404875-n7juj6b6go5ea1b4fmsenne0p7mgi3m7@developer.gserviceaccc	Can edit ▾
 858877404875@developer.gserviceaccount.com	Can edit ▾
 enhanced-killn-418@appspot.gserviceaccount.com	Can edit ▾
 ser72@owncloud.com	Can edit

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

## Email

ser72@owncloud.com

*Fill in an email address to enable password recovery*

Folder name	External storage	Configuration	Applicable
GoogleDrive	Google Drive	Client ID Client secret	None set

- 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.

Folder name	External storage	Configuration	Applicable
GoogleDrive	Google Drive	ooglleusercontent.com XtqTjg_irVIZpRY02Te Grant access	ser72 x

The following screen appears. Select Accept

# Project Default Service Account ▾

This app would like to:


 View and manage the files and documents in your Google Drive 

Project Default Service Account and Google will use this information in accordance with their respective terms of service and privacy policies.

Cancel

Accept

## External Storage

Folder name	External storage	Configuration	Applicable
 GoogleDrive	Google Drive	858877404875-n7juj6b nNOjXtqTjg_irVIZpRYX Access granted	<div>ser72 ✕</div>

Files

Activity

New

↑

Deleted files

Name	Size	Modified
documents	22.8 kB	2 hours ago
GoogleDrive	1009 B	2 months ago

**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.

External Storage

Folder name	External storage	Configuration	Applicable
OpenStackObjectStorage	OpenStack Object Storage	Username (required) Bucket (required) Region (optional for Opi API Key (required for R Tenanname (required for Password (required for Service Name (required	<div>URL of identity endpoint Timeout of HTTP reques None set</div>

- 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.

External Storage

Folder name	External storage	Configuration					Applicable
SMB	SMB / CIFS	URL	Username	Password	Share	Root	None set

- 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

External Storage

Folder name	External storage	Configuration					Applicable
 SMB	SMB / CIFS	192.168.1.58	ocmount	*****	/TedB	/	All Users x

---

**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.

---

 SMB 10 days ago

## ownCloud/WebDAV

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


External Storage

Folder name	External storage	Configuration					Applicable
ownCloud	ownCloud / WebDAV	URL	Username	Password	Root	<input checked="" type="checkbox"/> Secure https://	None set




- 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

External Storage

Folder name	External storage	Configuration				Applicable
 ownCloud	ownCloud / WebDAV	3/remote.php/webdav/	ser72	*****	/	Secure https://  ser72

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

	documents	22.8 kB	2 hours ago
	music	3.6 MB	2 hours ago
	ownCloud	?	21 days ago

## SFTP

Mounts a folder on a remote SSH server.

External Storage



Folder name	External storage	Configuration				Applicable
SFTP	SFTP	URL	Username	Password	Root	None set

- 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

External Storage

Folder name	External storage	Configuration				Applicable
 SFTP1	SFTP	192.168.1.68	yogi	*****	/	 yogi

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

New				Deleted files
<input type="checkbox"/>	Name	Size	Modified	
	SFTP1			? 3 months ago

## iRODS

Mounts a folder on a iRODS server.

External Storage

Folder name	External storage	Configuration						Applicable	
iRODS	iRODS	Host	Port	<input checked="" type="checkbox"/> Use ownCloud login	Username	Password	Authentication Mode	Zone	None set

- 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.

---

## 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.

## 4.6 Sharing

ownCloud allows users to share files and folders with both other ownCloud users, as well as publicly via a link. Sharing may be done either via the web interface or via the Sharing API. This document will discuss the web interface and sharing. For information on the Sharing API, please refer to that document.

### 4.6.1 Configuration

#### Enable the app

The Share Files app is enabled by default. To verify this and/or disable this functionality, navigate to the APPS page in the web interface and select “Share Files”



**Share Files 0.3.5** Internal App

File sharing between users

AGPL-licensed by Michael Gapczynski

**Disable****Configuring the permitted functionality**

There are several functions which may be enabled or disabled at the admin's discretion. To configure these functions, navigate to the Admin page in the web interface and scroll to the "Sharing" section.

**Sharing**☒ Enable Share API*Allow apps to use the Share API*☒ Allow links*Allow users to share items to the public with links*☒ Allow public uploads*Allow users to enable others to upload into their publicly shared folders*☒ Allow resharing*Allow users to share items shared with them again*☒ Allow users to share with anyone☐ Allow users to only share with users in their groups☒ Allow mail notification*Allow user to send mail notification for shared files*

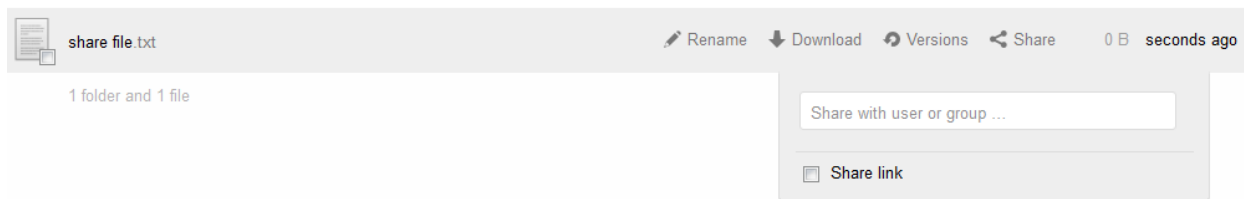
The above parameters are self-explanatory and may be enabled or disabled as required.

**4.6.2 Utilization**

An ownCloud user may share files or entire folders with other individuals. The method of sharing is the same.

**Share with another ownCloud user**

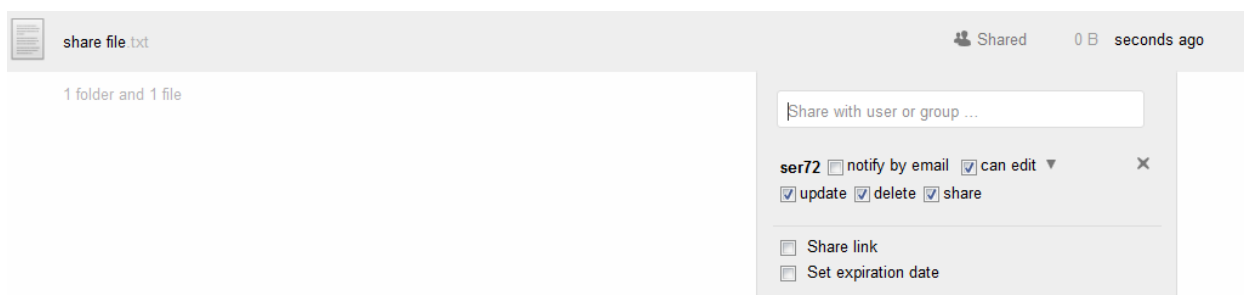
Hover over the line containing the file or folder to be shared. And then select Share. The following dialog appears.



In the “share with user or group” text box, enter the user to share with, or the group to share with. Note, as letters are typed, the users and groups who match the typed string will appear. As a short cut, simply select the desired user/group

### Setting Permissions on the share

When a file or folder is shared, the permissions may be modified stating what the other user may do with the file/folder. Once the share with user or group has been entered, the dialog will expand as follows.



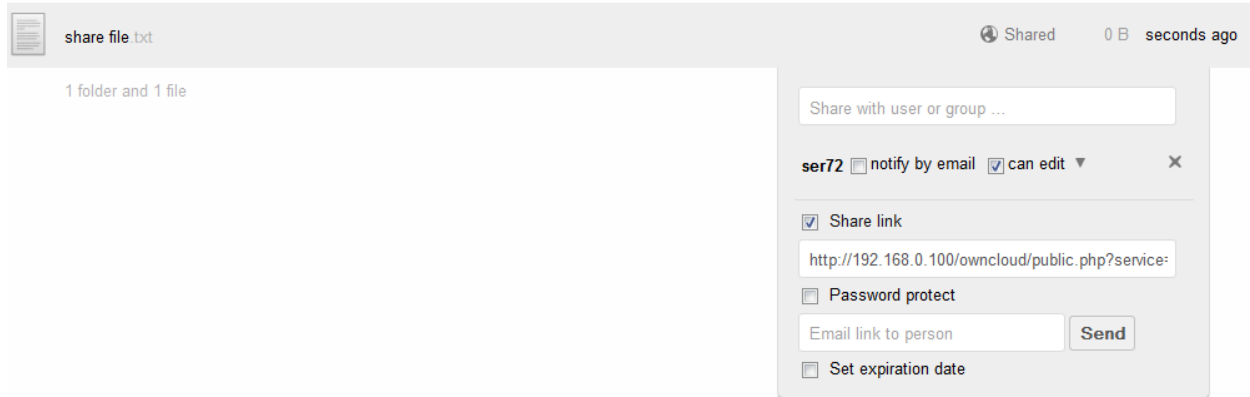
If the shared with user has an email address configured, the sharing user may choose to notify the shared with user of the share. To do this, select the checkbox next to “notify by email”. It is important to note, this checkbox appears whether or not the shared with user has email configured.

When sharing a file/folder, it is obvious that the file may be viewed or downloaded. However, if the shared with user is allowed to modify the file in any way, the “can edit” box should be checked. The sharing user may allow the shared with user to update, delete, or re-share the file. Note, by default, all permissions are set.

It should be noted that when sharing a folder, an additional permission, “create” is available. When enabled, the shared to users can create new files and folders within the shared folder.

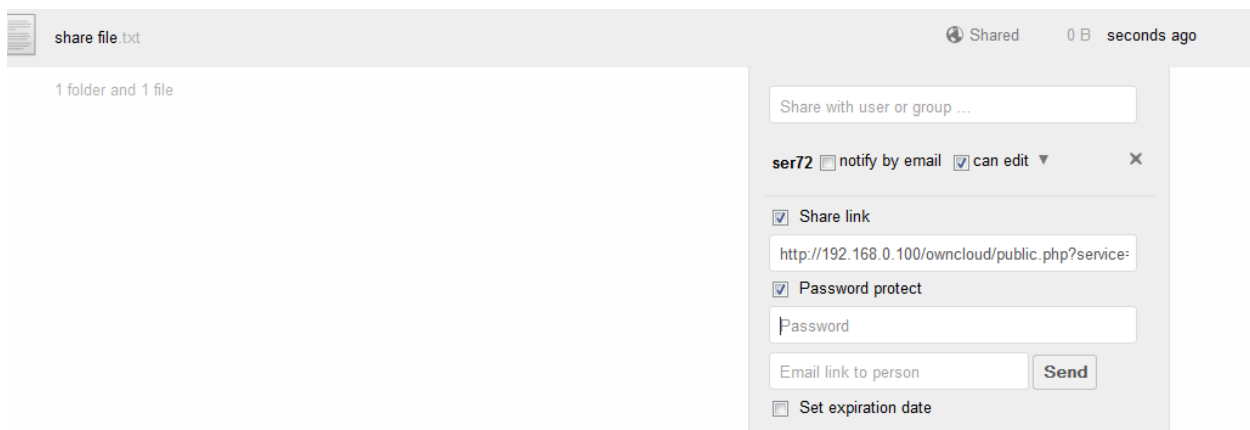
### Share as a link

In the event that an ownCloud user desires to share a file with a non-ownCloud user, the file may be shared as a link. To share as a link, hover over the line containing the file and select Share. Then select the check box next to “Share link”



The link is then shown in the text box. The link may be copied and pasted into an email and sent to the other user. Alternatively, if email is setup within ownCloud, the sharing user may input the shared with user's email address in the text box and select send. The shared user will then receive an email from the ownCloud server with the link.

There may be circumstances where the file owner wants some security on the link so it cannot be accessed by anyone other than the desired person. To do this, the link may be Password Protected by selecting the "Password Protect" checkbox and entering a desired password.



## Setting Expiration Date

ownCloud allows the sharing user to expire a file or folder share at a given date. This expiration applies to all shares within the folder as well. So, if a user expires a folder share, all files within the folder will no longer be shared.

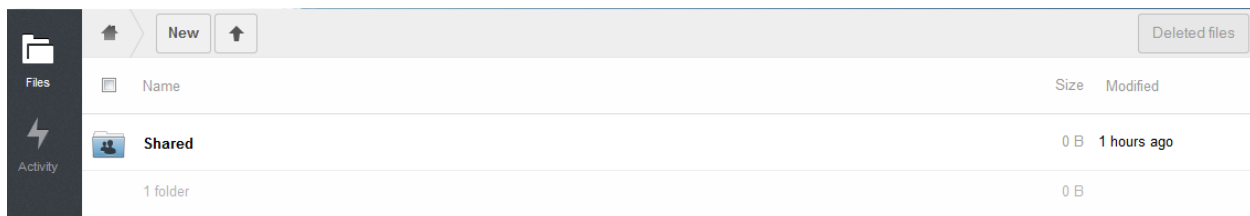
To set an expiration, select the Set expiration date checkbox and then select the expiration date. (The calendar will appear when the mouse is clicked in the Expiration date text box.



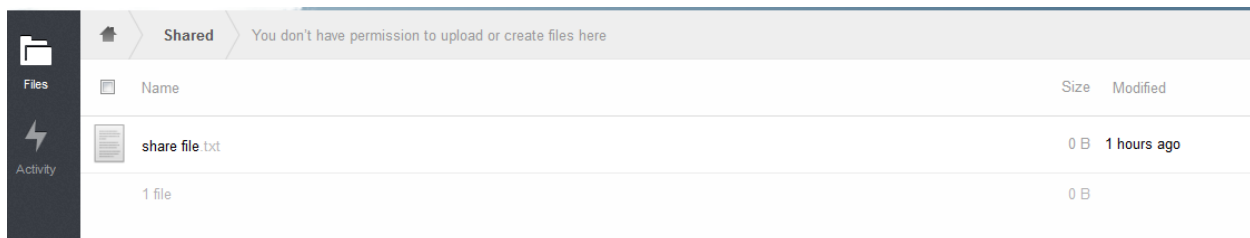
## 4.6.3 Retrieving shared data

### User/group shares

When the shared with user logs into their ownCloud instance they will see a “Shared” folder.



Within that Shared folder are all files and folders which have been shared to that user by any other ownCloud user.



### Share as link

To view a file/folder which was shared as a link, simply enter the link into the browser.

To download the file/folder select the Download button on the upper right of the page. If this link was password protected, the user will be prompted with a password prior to this page appearing.



## 4.7 Deleted Files

The ownCloud server stores deleted files in a temporary area in the event that the file was inadvertently deleted and/or needs to be restored.

### 4.7.1 Expiry of deleted files

There are two instances in which ownCloud will automatically permanently remove a deleted file.

#### Disk Utilization

To prevent a user from running out of disk space, the ownCloud deleted files app will not utilize more than 50% of the currently available free space for deleted files. If the deleted files exceed this limit, ownCloud deletes the oldest files until it gets below this limit.

#### Age

By default, deleted files remain in the trash bin for 30 days. This can be configured using the `trash-bin-retention-obliteration` parameter in the `config.php` file. Files older than the configured value (or default 30 days) will be permanently deleted. ownCloud checks the age of the files each time a new file is moved to the deleted files bin.

### 4.7.2 Configuration and storage

#### Configuration

By default, the ownCloud deleted files app is enabled. To verify or disable, navigate to the apps page and select Deleted Files.

**Deleted files** 0.4 Internal App

ownCloud keeps a copy of your deleted files in case you need them again. To make sure that the user doesn't run out of memory the deleted files app manages the size of the deleted files for the user. By default deleted files stay in the trash bin for 30 days. ownCloud checks the age of the files every time a new file gets moved to the deleted files and remove all files older than 180 days. The user can adjust this value in the config.php by setting the "trashbin\_retention\_obligation" value. Beside that the deleted files app take care to never use more than 50% of your currently available free space. If your deleted files exceed this limit ownCloud deletes the oldest versions until it meets the memory usage limit again.

AGPL-licensed by Björn Schiesse

Disable

## Storage

Once a file has been deleted by the user, it is moved to the `~/data/<user>/files_trashbin/files` folder.

```
root@S3FS:/var/www/owncloud/data/yogi/files_trashbin# ls
files keyfiles share-keys versions
root@S3FS:/var/www/owncloud/data/yogi/files_trashbin#
```

The remaining directories retain information on encryption key files, and versions.

### 4.7.3 Utilization

The deleted files app, when enabled, automatically moves deleted files to the Deleted Files folder and leaves them available for restore or permanent deletion

#### Delete a file

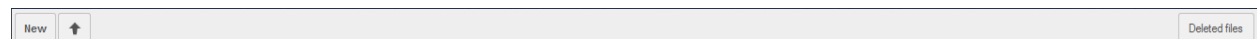
To delete a file, either select the file check box and select Delete on the upper right of the screen, or the “x” to the right of the file.



To delete multiple files simultaneously, select the check box on all the desired files, then select Delete on the upper right of the screen.

#### View Deleted Files

To view a list of the deleted files, select the Deleted files button on the upper right of the browser.



Once selected, a list of all deleted files will appear.

### Restore files

As with deleting files, there are two ways to restore a file. Either select the check box next to the file (or for bulk restore – files) and select restore on the upper right. Or hover over the file and select restore .

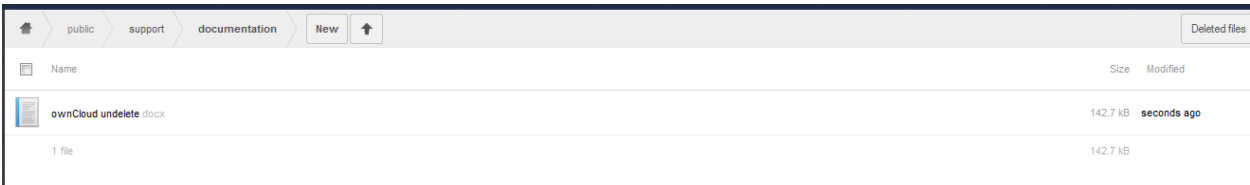


### Permanently Delete Files

Files in the Deleted Files folder can be permanently deleted. To do this, either select the check box next to the file (or for bulk deletion – files) and select Delete in the upper right corner. Or hover over the file and select the “x”.

### Nested files and restore

If, for instance, the directory structure within ownCloud is `~/public/support/documentation/ownCloud undelete` .



### Delete entire directory structure

When the public folder is deleted, all child folders/files will also be deleted.



Suppose the file “ownCloud undelete.docx” was still required. A restore of the file will place it in the ‘root’ directory of the Files folder.

### Delete only the file

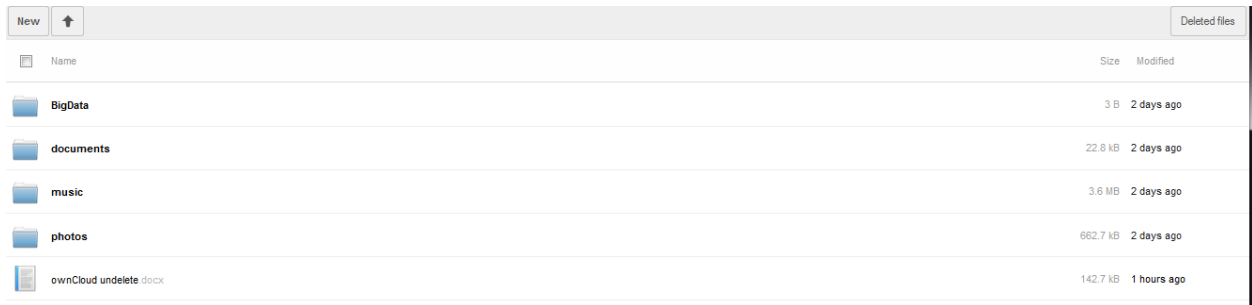
If the file “ownCloud undelete.docx” was accidentally deleted, it may be restored following the steps described in section . The restore will place the file back into the directory structure from where it came.

## Delete the file then the directory structure

If the file “ownCloud undelete.docs” is deleted, then the entire directory structure is deleted, the file will appear in the ‘root’ of the Deleted Files folder.



A restore of “ownCloud undelete.docx” will place it in the user’s ownCloud root directory.



## Shared files and restore

When a shared file is deleted, the file will be deleted from the shared to user as well. Upon restore of the file by the file owner, the file is no longer shared.

## Restore files with Versions

When a file which has versions has been deleted, and then restored, the versions will exist upon restoration.

## 4.8 Versions

The ownCloud Versions app maintains older versions of a file which have been modified by an ownCloud user.

### 4.8.1 Expiry of versions

The versions app expires old versions automatically to make certain that the user doesn’t run out of space. The following algorithm is used to delete old versions:

- ownCloud keeps one new version every 2 seconds for the first 10 seconds
- ownCloud keeps one new version every minute for the first hour
- ownCloud keeps one new version every hour for the first 24 hours
- ownCloud keeps one new version every day for the first 30 days
- ownCloud keeps one new version every week thereafter.

The versions are adjusted along this algorithm every time a new version is created.



## 4.8.2 Space limitations

In addition to the expiry of versions, ownCloud's versions app makes certain never to use more than 50% of the user's currently available free space. If stored versions exceed this limit, ownCloud will delete the oldest versions first until it meets this limit.

## 4.8.3 Configuration and storage

### Configuration

By default, the ownCloud versions app is enabled. To verify or disable, navigate to the apps page and select Versions.

#### Versions 1.0.3 Internal App

ownCloud supports simple version control for files. The versioning app expires old versions automatically to make sure that the user doesn't run out of space. Following pattern is used to delete old versions: For the first 10 seconds ownCloud keeps one version every 2 seconds; For the first hour ownCloud keeps one version every minute; For the first 24 hours ownCloud keeps one version every hour; For the first 30 days ownCloud keeps one version every day; After the first 30 days ownCloud keeps one version every week. The versions are adjusted along this pattern every time a new version gets created. Beside that the version app takes care to never use more than 50% of the users currently available free space. If the stored versions exceed this limit ownCloud deletes the oldest versions until it meets the memory usage limit again.

AGPL-licensed by Frank Karlitschek

[Disable](#)

### Storage

Previous versions of files are stored in the `data/<user>/files_versions` folder.

```
root@S3FS:/var/www/owncloud/data/yogi# ls
files          files_trashbin  gallery        thumbnails
files_external files_versions  lucene_index
root@S3FS:/var/www/owncloud/data/yogi#
```

This directory is automatically created when the first file version is created.

The naming convention of the files in this directory are:

`<file_name>.v<unix_timestamp>`

```
root@S3FS:/var/www/owncloud/data/yogi/files_versions# ls
versionTest.txt.v1384935260
root@S3FS:/var/www/owncloud/data/yogi/files_versions#
```

## 4.8.4 Utilization

The versions app, when enabled, automatically creates a new version each time an existing, non-empty file is saved.

### Initial File creation

Create a file on the ownCloud server – either directly via the browser, an upload, or a file sync from the client. In this example, the file `versionText.txt` was created via the web browser.



versionTest.txt

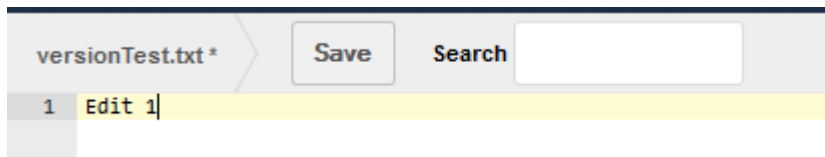
0 B seconds ago

### Note

Since this file was initially created via the web browser, it is an empty file.

### Edit the file

Edit the file through the web browser.



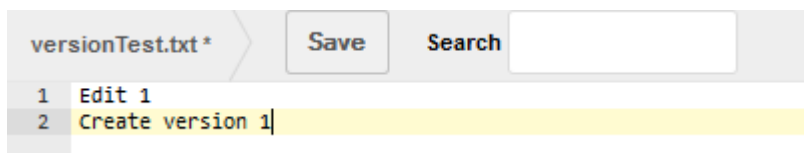
And save.

### Notes

Editing an empty file, as in this instance, for the first time, does not create a new version.

### Create version

Edit the non-zero byte file either via the web browser, via the sync client or via an upload.



And save.

### Notes

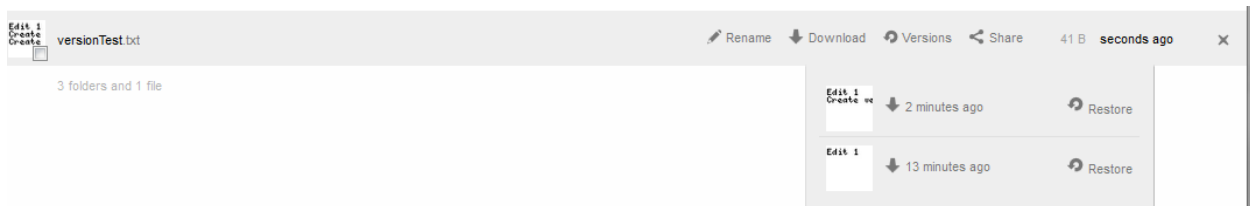
When performing an upload of a file which already exists on the server, a dialog box appears asking whether to keep the existing file or the new file.



In order to create a version, the New File must be selected. If the Existing File is selected, the file is not replaced hence no new version is created.

## Accessing Versions

In order to view what versions exist for a given file, hover over the line containing the file and select Versions on the right side.



In this case, there are two previous versions of versionTest.txt . The thumbnail and creation time can be used to identify what was in the file in that version.

## Reverting to a different version

To revert to a previous version, simply select the restore next to the version as seen in .

## Notes

In the instance where a file is reverted to another version, the “existing” version of the file is versioned and can be reverted at a later date.

## Sharing and Versions

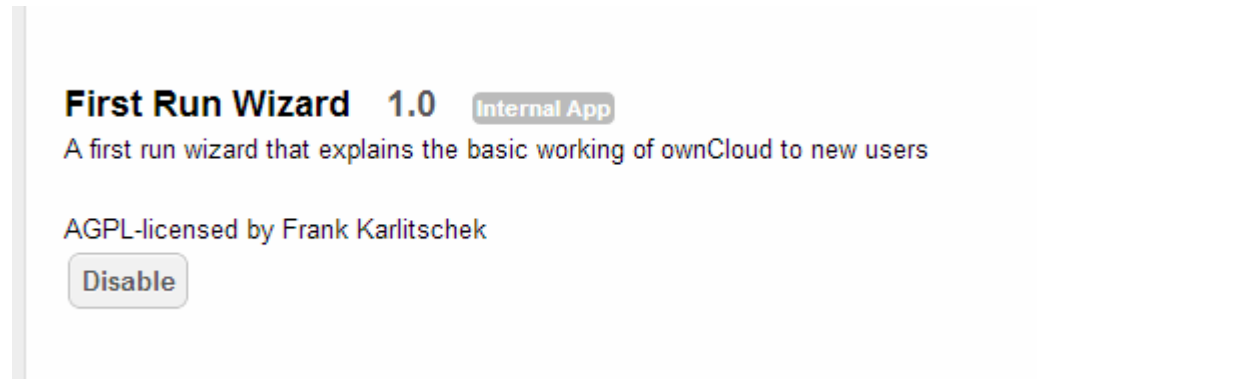
If user-A shares a file with user-B, user-B may revert the file to any previous version. If user-B modifies the file, a new version is created.

## 4.9 First Run Wizard

The ownCloud First Run Wizard is a welcome screen which links users to the various ownCloud utilities such as the Sync Client, Mobile Apps, and WebDav

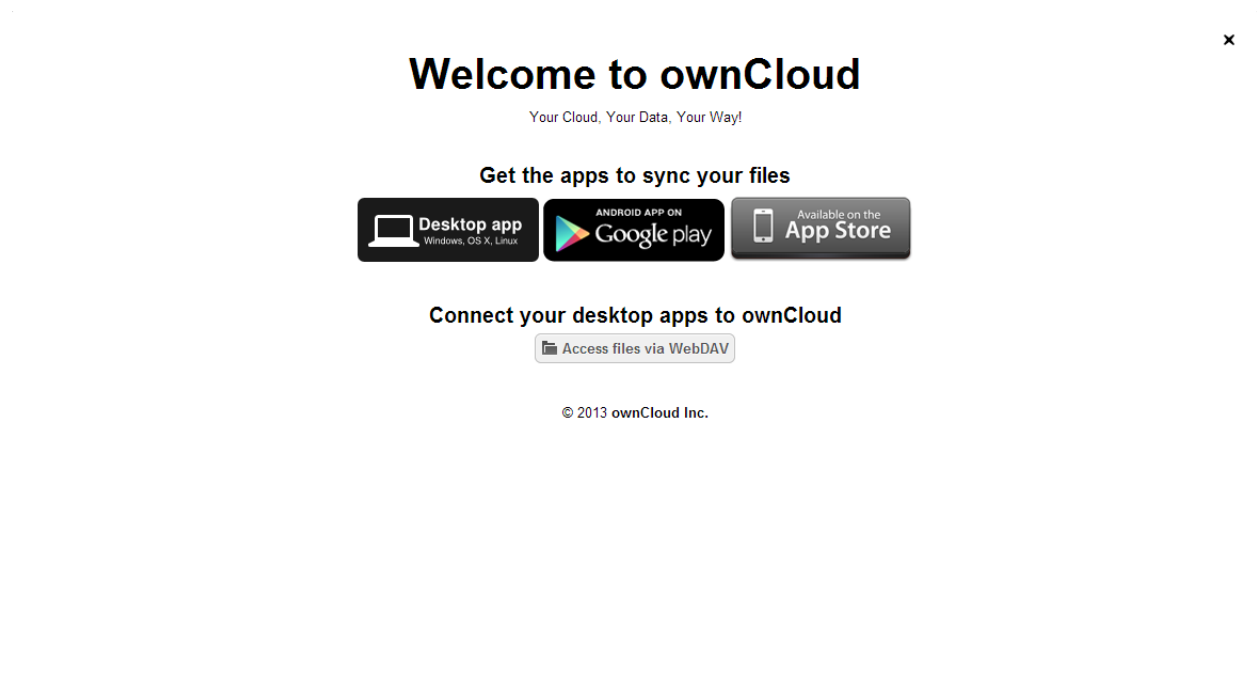
### 4.9.1 Configuration

The First Run Wizard app is enabled by default. To verify or disable this app, navigate to the apps page and select “First Run Wizard”



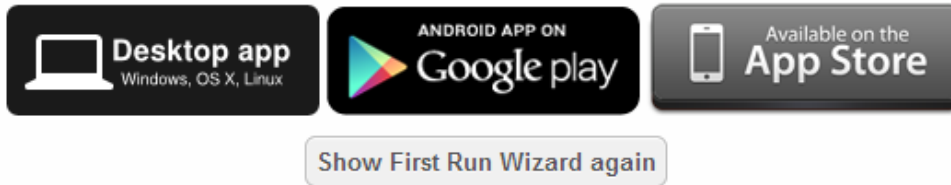
### 4.9.2 Usage

When enabled, a user entering the ownCloud web browser for the first time will have the Welcome screen popup.



To close the window, select the ‘x’ in the upper right corner. If the user would like to bring up this window at a later time, they may do so by navigating to the Personal menu selection and select “Show First Run Wizard again”

## Get the apps to sync your files

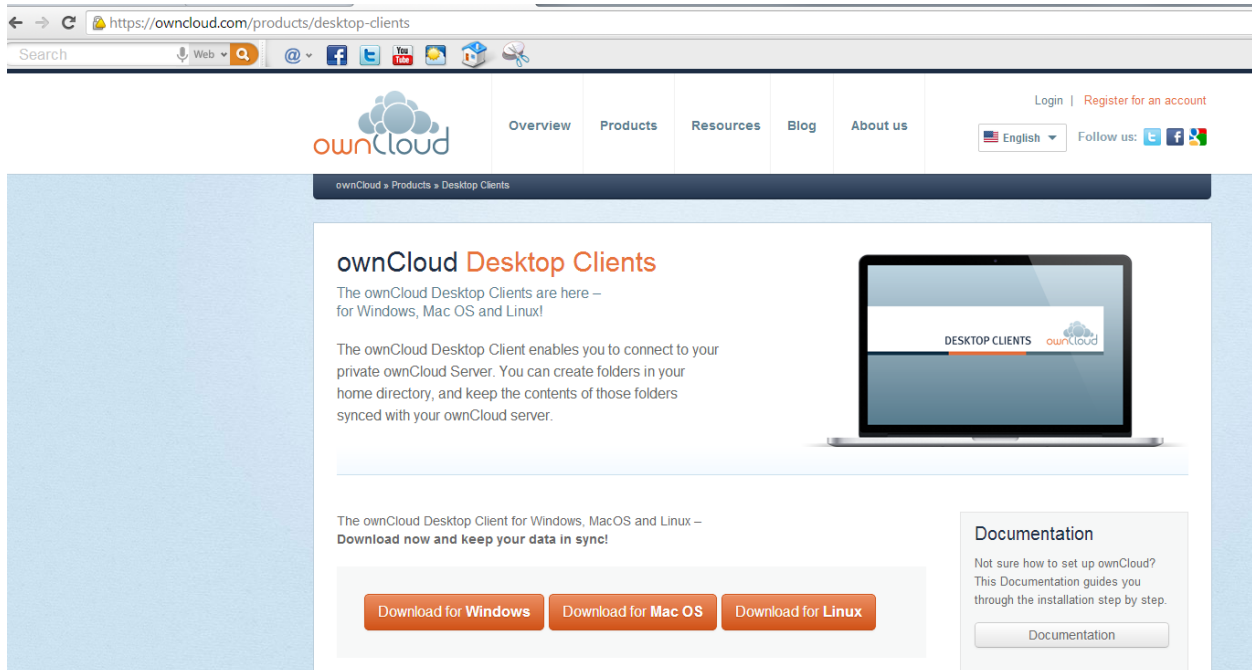


### 4.9.3 Links

The wizard contains links for the Desktop app, Android App on Google Play, and iOS on the App store, as well as how to access files via WebDav.

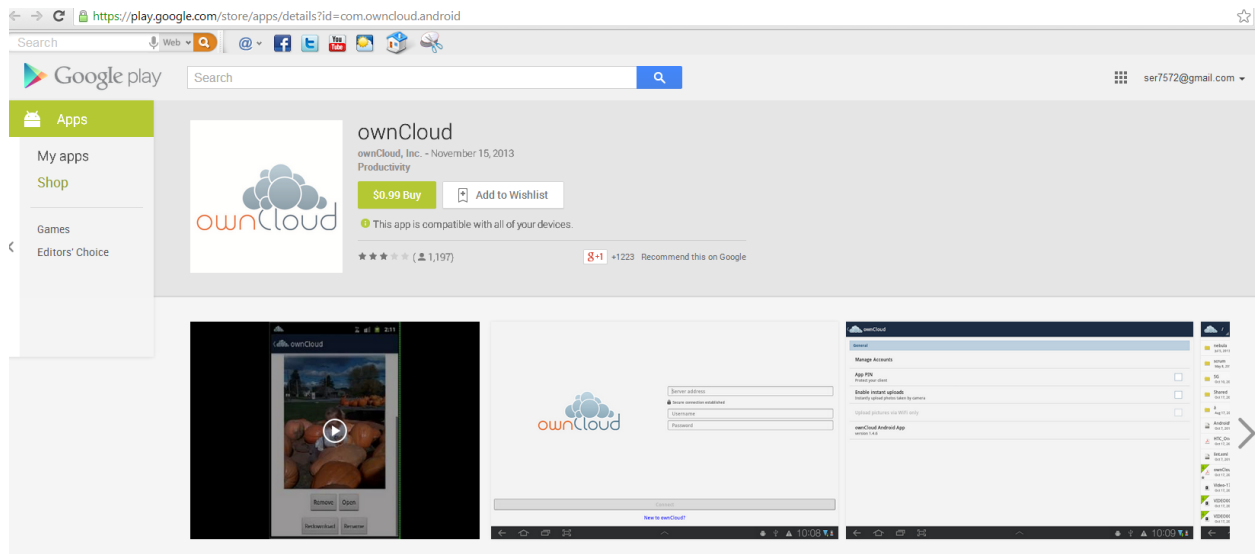
#### Desktop App

Selecting the Desktop App link will bring the user to ownCloud's web site to download the client.



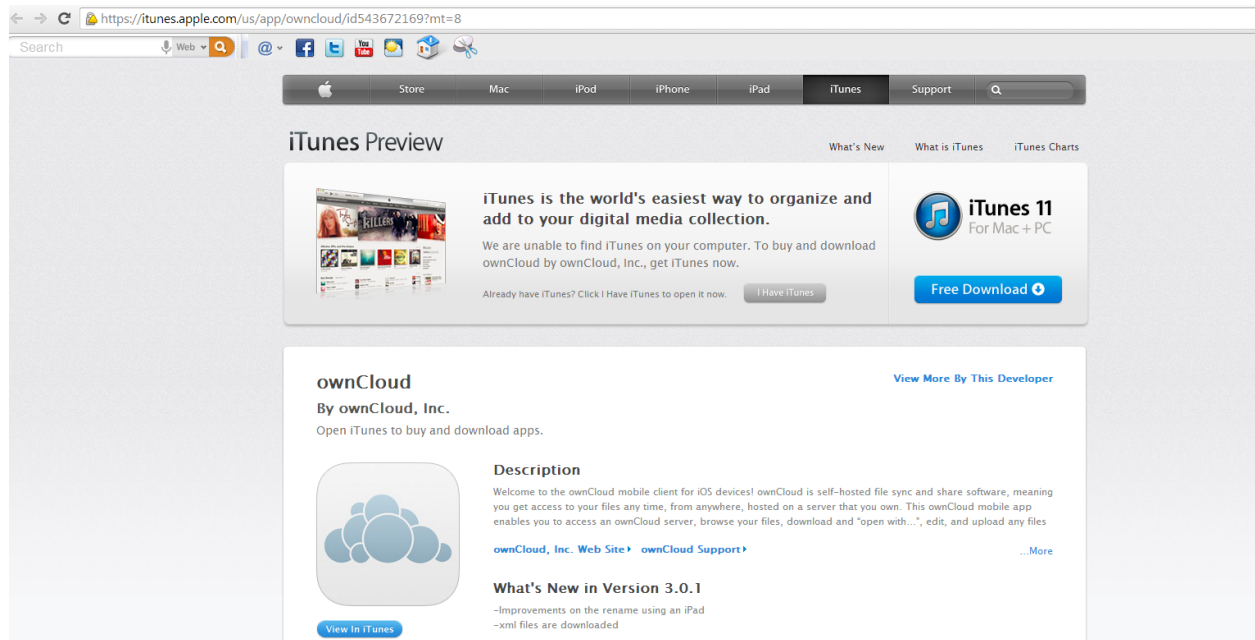
#### Android App

Selecting the Android App will load the web page to purchase the android client from the Google Play store.



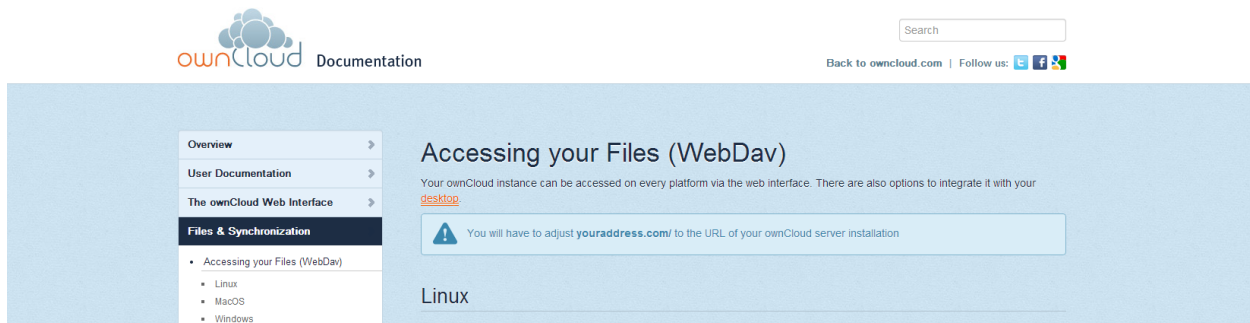
### iOS App

Selecting the iOS App loads the iTunes web page to obtain the ownCloud iOS mobile client.



### Access files via WebDAV

This link brings up a link with documentation on how to access files on the ownCloud server using WebDAV.



## 4.10 LDAP user and group backend

ownCloud ships an LDAP backend, which allows full use of ownCloud for user logging 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 features.

### 4.10.1 Configuration

#### Enable LDAP app

From the APPs page, select “LDAP user and group backend” and select enable

#### **LDAP user and group backend** 0.4.1 Internal App

Authenticate users and groups by LDAP respectively Active Directory. This app is not compatible with the WebDAV user backend.

AGPL-licensed by Dominik Schmidt and Arthur Schiwon

**Disable**

#### Configuring LDAP

The configuration of the LDAP feature is performed on the Admin page of the ownCloud web browser. The configuration follows a wizard-like approach split into four tabs. The first tab must be completed correctly to allow access to subsequent tabs. Although the configuration in the remaining tabs is detected automatically, it should be reviewed by the admin to verify correctness. An indicator exists on the all pages to show whether the configuration is valid, incomplete, or incorrect.

Configuration settings are saved automatically when the cursor is no longer focused on the input element.

#### Server Tab

The server tab contains basic information on the LDAP server. It makes certain that ownCloud can connect to the desired LDAP and read data from it. At a minimum, the admin must provide a hostname for the LDAP server. If

anonymous access to the LDAP server is not allowed, the admin will be required to enter an account Distinguished Name (DN) and password. ownCloud will auto-detect the port and base DN.

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

**Host** The hostname of the LDAP server. It can also be a `ldaps://` URI. It is possible to pass a port number which will speed up port detection. This is especially useful if a custom port is used. ownCloud will subsequently move the port value to the port field.

Examples:

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

**Port** This is the port which ownCloud should utilize to connect to the LDAP server. Upon initial configuration, this field is disabled. ownCloud will auto-detect the port if it is running on a standard port (389). After ownCloud detects the port, the field will be enabled for admin input. A successfully discovered port will be inserted by ownCloud.

**User DN** The name as DN of a user who is able to do searches in the LDAP directory. For anonymous access, leave this field blank. It is recommended to have a special system user for ownCloud. This information is provided by the LDAP admin.

**Password** The password for the user given above. For anonymous access, leave this blank. This information is provided by the LDAP admin.

**Base DN** The base DN of the LDAP from where all users and groups can be reached. It is possible to provide separated base DN's for users and groups in the advanced tab. This is a mandatory field. ownCloud will attempt to determine the proper value of this field based on the provided User DN or Host values.

The screenshot shows the 'Server' tab of the LDAP configuration wizard. At the top, there are tabs for 'Server', 'User Filter', 'Login Filter', and 'Group Filter', along with 'Advanced' and 'Expert' options. The 'Server' tab is active. It features a dropdown menu labeled '1. Server' with a 'Delete Configuration' button next to it. Below this, there are four input fields: the first contains '192.168.1.50' and the second contains '389'; the third contains the LDAP admin DN 'cn=ownCloudAdmin,ou=ownCloudAccess,dc=owncloud1,dc=com'; and the fourth contains the base DN 'dc=owncloud1,dc=com'. At the bottom, there is a green status indicator labeled 'Configuration OK', a 'Continue' button, and a 'Help' link.

## User Filter

The settings in the User Filter tab determine which LDAP users will appear and will be able to log into ownCloud. This may be configured using the wizard or entered via a raw LDAP filter.



**Only those object classes** ownCloud will determine the available object classes. ownCloud will automatically select the object that contains the highest number of users. It is possible to select multiple object classes.

**Only from those groups** This is used if the LDAP server supports the member-of-overlay in LDAP filters. It allows the admin to define the users from one or more certain groups that are allowed to appear and log into ownCloud. No value is selected by default. It is possible to select multiple groups.

If the LDAP server does not support member-of-overlay in LDAP filters, the field is disabled.

**Edit raw filter instead** Selecting this text will toggle the filter mode. Instead of the wizard's assistance, the admin may enter the raw LDAP filter in this field.

Example:

```
objectClass=inetOrgPerson
```

**x users found** Indicates the approximate number of users allowed to access ownCloud. This number will update after any changes made to the LDAP configuration.

The screenshot shows the 'User Filter' tab of the LDAP configuration wizard. At the top, there are tabs for 'Server', 'User Filter', 'Login Filter', and 'Group Filter', along with 'Advanced' and 'Expert' options. The main area is titled 'Limit the access to ownCloud to users meeting this criteria:'. It contains two dropdown menus: 'only those object classes:' with 'person' selected, and 'only from those groups:' with 'ownCloudUsers' selected. Below these is a link 'Edit raw filter instead'. At the bottom left, it says '37 users found'. At the bottom right, there is a status 'Configuration OK' with a green dot, and buttons for 'Back', 'Continue', and 'Help'.

## Login Filter

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

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

**LDAP Email Address** If checked, the login credentials will be compared to an email address in the LDAP directory. ownCloud will examine the `mailPrimaryAddress` and `mail` attributes in the LDAP for the email address.

**Other attributes** This field allows the admin to select additional attributes for comparison. The list is generated automatically based on the attributes contained in the user object of the LDAP server.

**Enter raw filter instead** Selecting this text will toggle the filter mode. Instead of the wizard's assistance, the admin may enter the raw LDAP filter in this field.

The %uid placeholder will be replaced with the login name entered by the user upon login.

Examples:

- Username only:

```
uid=%iud
```

- Username or email address:

```
( | (uid=%uid) (mail=%uid) )
```

The screenshot shows the 'Login Filter' tab of the LDAP configuration wizard. The interface has a dark blue header with tabs: 'Server', 'User Filter', 'Login Filter' (selected), and 'Group Filter'. On the right of the header are 'Advanced' and 'Expert' buttons. The main content area is light gray and contains the text 'What attribute shall be used as login name:'. Below this are three options: 'LDAP Username:' with a checked checkbox, 'LDAP Email Address:' with an unchecked checkbox, and 'Other Attributes:' with a text input field containing 'Select attributes' and a dropdown arrow. At the bottom left of the main area is a link '⌵ Edit raw filter instead'. At the bottom center is a status bar showing 'Configuration OK' with a green dot, and 'Back', 'Continue', and 'i Help' buttons.

## Group Filter

The settings in this tab determine which groups will be available in ownCloud. This tab does not restrict logins in any manner as that was handled in the prior tabs. It is possible to enter a raw LDAP filter as well.

By default, there are no groups available in ownCloud. The admin must enable this manually.

**Only those object classes** ownCloud will automatically determine which object classes are available in the LDAP. ownCloud will only list object classes that return at least one group object. It is possible to enter multiple object classes.

**Only from those groups** This setting allows the admin to select which groups are available within ownCloud. ownCloud will generate a list of available groups found in the LDAP server for the admin to select. It is possible to enter multiple groups.

**Edit raw filter instead** Selecting this text will toggle the filter mode. Instead of the wizard's assistance, the admin may enter the raw LDAP filter in this field

**Y groups found** Indicates the approximate number of groups available in ownCloud. This number will update after any changes made to the LDAP configuration.

Server User Filter Login Filter **Group Filter** Advanced Expert

Limit the access to ownCloud to groups meeting this criteria:

only those object classes: group

only from those groups: ownCloudUsers

[Edit raw filter instead](#)

1 group found

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

### Advanced Tab

The LDAP Advanced settings section allows the admin to define less common options. These options are not required for a working connection however, they can have a positive effect on the performance.

The Advanced Settings tab has three sections

- Connection settings
- Directory settings
- Special attributes

### Connection Settings

**Configuration Active** Allows the admin to enable or disable the current configuration. A disabled configuration will not connect to the LDAP server.

By default, this is disabled. It is enabled automatically, when using the wizard and configuration is valid and tests successfully.

**Backup (Replica) Host** This is used to define a backup LDAP server. ownCloud automatically attempts to connect to the backup server when the primary server cannot be accessed. It is important that the backup server is an exact replica of the primary server as all the object UUIDs must match.

**Backup (Replica) Port** This identifies the port on which ownCloud will connect to the backup LDAP server. If no port is provided, ownCloud will utilize the same port as the primary LDAP server.

**Disable Main Server** This is used to disable the primary LDAP server so ownCloud will connect only to the backup server. This can be useful for planned maintenance on the primary server.

**Case insensitive LDAP server (Windows)** Check this if the LDAP server is running on a windows host. Not usually necessary.

**Turn off SSL certificate validation** Disables the check for a valid SSL certificate. It is recommended to use for testing only if needed, but not use in production.

**Cache Time-To-Live** ownCloud caches the information it receives from the LDAP server. This is necessary as the ownCloud server attempts to validate the user with every page request or WebDAV interaction. This time is in seconds.

Note if it is required to have the most up-to-date information from the LDAP, it is recommended not to turn off the cache totally, however, to define a lifetime of a small duration (15 seconds)

The screenshot shows the 'Advanced' settings tab in ownCloud. The 'Connection Settings' section is expanded, showing the following options:

- Configuration Active: ☒
- Backup (Replica) Host:
- Backup (Replica) Port:
- Disable Main Server: ☐
- Case insensitive LDAP server (Windows): ☐
- Turn off SSL certificate validation: ☐
- Cache Time-To-Live:

Below the Connection Settings section are two collapsed sections: 'Directory Settings' and 'Special Attributes'. At the bottom of the settings panel are three buttons: 'Save', 'Test Configuration', and 'Help'.

## Directory Settings

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

**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.

**User Search Attributes** These attributes are used when a search for users with a search string is done. This happens in the share dialogue. By default the user display name attribute as specified above is used. Multiple attributes can be given, one in each line.

Note: if an attribute is not available for a given user object, the user will neither be listed nor able to login.

**Group Display Name Field** The attribute that should be used as an ownCloud group name. ownCloud allows a limited set of characters (regex notation):

```
[a-zA-Z0-9, -_@]
```

Every other character will be replaced in ownCloud. Once a group name is assigned, it will not be changed.

**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.

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

**Group Member association** The attribute that is used to indicate group memberships.

ServerUser FilterLogin FilterGroup FilterAdvancedExpert

Connection Settings

Directory Settings

User Display Name Fielddisplayname

Base User TreeOU=ownCloudAccess,DC=owncloud1,DC=com

User Search AttributesOptional; one attribute per line

Group Display Name Fieldsamaccountname

Base Group TreeOU=ownCloudAccess,DC=owncloud1,DC=com

Group Search AttributesOptional; one attribute per line

Group-Member associationmember (AD)

Special Attributes

SaveTest Configurationi Help

Special Attributes

**Quota Field** This field is used to set a LDAP attribute to define the user quota. The attribute should retain a readable value, for example:

2 GB

**Quota Default** This is used to override the ownCloud default quota for LDAP users who do not have an attribute set in the above parameter.

Example:

15GB

**Email Field** ownCloud will read the attribute configured here and configure the user’s email.

**User Home Folder Naming Rule** By default, ownCloud creates a user directory which contains all files and meta data based on the ownCloud user name. To override this setting and name it after a different attribute, configure that attribute here. The attribute can also return an absolute path (such as /mnt/storage43/alice).

## Expert Tab

### 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 (regex notation):

```
[a-zA-Z0-0_.@-]
```

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 back ends (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 behavior can be overridden.

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

### 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 from above.

You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behavior. Changes will have effect only on newly mapped (added) LDAP users and groups. 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).

### 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 an 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 is not used for identification. If the DN changes, the change will be detected by ownCloud by checking the UUID value.

The same is valid for groups.

The internal ownCloud name is used all over in ownCloud. Clearing the mappings will have leftovers everywhere. Never clear the mappings in a production environment. Only clear mappings in a test or experimental stage.

**Internal Username**

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

Internal Username Attribute:

**Override UUID detection**

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

UUID Attribute for Users:

UUID Attribute for Groups:

**Username-LDAP User Mapping**

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

Clear Username-LDAP User Mapping

Clear Groupname-LDAP Group Mapping

Save Test Configuration *i* Help

## 4.11 File Viewers

ownCloud provides apps which allow users to view/edit text files and view images which exist on the ownCloud server.

### 4.11.1 Configuration

#### Text Editor

The Text Editor App allows users to view and edit text files. To enable this app, navigate to the Apps page and select “Text Editor” then Enable.

**Text Editor** 0.3 Internal App

Simple plain text editor based on Ace editor.

AGPL-licensed by Tom Needham

**Enable**

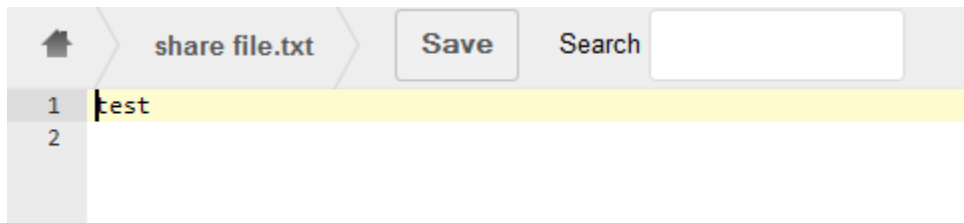
## Image Viewer

The image viewer app allows users to preview image files within the ownCloud web interface. To enable, navigate to the Apps page and select “Image Viewer” then enable.



### 4.11.2 Utilization

When the apps are enabled, select the file name on the web interface Files tab and either the text editor or the image will appear.



When the apps are disabled, selecting the file name on the web interface will prompt the user to download the specified file.



## **MAINTENANCE**

### **5.1 Backing up ownCloud**

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

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

#### **5.1.1 Backup Folders**

Simply copy your config and data folder (or even your whole ownCloud install and data folder) to a place outside of your ownCloud environment. You could use this command:

```
rsync -Aax owncloud/ owncloud-dirbkp_`date +%Y%m%d` \/
```

#### **5.1.2 Backup Database**

##### **MySQL**

MySQL is the recommended database engine. To backup MySQL:

```
mysqldump --lock-tables -h [server] -u [username] -p[password] > owncloud-sqlbkp_`date +%Y%m%d` \.bak
```

##### **SQLite**

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

##### **PostgreSQL**

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

## 5.2 Updating ownCloud

---

**Note:** If you have installed ownCloud from a repository, your package management should take care of it. Probably you will need to look for compatible third party applications yourself. **Always do backups anyway.**

---

### 5.2.1 Update

Updating means updating ownCloud to the latest *point release*, e.g. ownCloud 6.0.0a → 6.0.2. This procedure uses the ownCloud updater plugin called “Updater”: it’s an internal application already present in your ownCloud installation.

To update ownCloud, follow those steps:

1. Make a backup of the ownCloud folder and the database.
2. Make sure that updater plugin is enabled.
3. Navigate to the ‘Admin’ page.
4. Click ‘Update’.
5. Refresh the page with 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).

### 5.2.2 Upgrade

Upgrade is to bring an ownCloud instance to a new *major release*, e.g. ownCloud 5.0.14a → 6.0.2. Always do backups anyway.

To upgrade ownCloud, follow those steps:

1. Make sure that you ran the latest point release of the major ownCloud version, e.g. 5.0.14a in the 5.0 series. If not, update to that version first (see above).
2. Make a backup of the ownCloud folder and the database.
3. Download the latest version to the working directory:

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

4. Deactivate all third party applications.
5. Delete everything from your ownCloud installation directory, except data and config.
6. Unpack the release tarball in the ownCloud directory (or copy the files thereto). Assuming that your installation directory is called ‘owncloud’ and that it’s inside your working directory, you could execute this command:

```
tar xvj owncloud-latest.tar.bz2
```

7. Set the permissions properly
8. With the next page request the update procedures will run.
9. If you had 3rd party applications, check if they provide versions compatible with the new release. If so, install and enable them, update procedures will run if needed.

## 5.3 Restoring ownCloud

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

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

### 5.3.1 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/
```

### 5.3.2 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] < owncloud-sqlbkp.bak
```

#### SQLite

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

#### PostgreSQL

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

## 5.4 Migrating ownCloud Installations

To migrate an ownCloud install, follow those steps:

1. Backup data/config folders and your database (look at “Backing Up ownCloud”)
2. Move your data
3. Restore your data/config folders and your database (look at “Restore ownCloud”)
4. Update config.php of any changes to your database connection



## THE CONFIGURATION FILE

### 6.1 Default Parameters

The following parameters are automatically configured by ownCloud upon the initial admin login. These parameters are mandatory for the ownCloud instance to operate properly.

Parameter	Format	Description
<b>Debug</b>	<code>define("DEBUG", true);</code>	When set to true, additional information is written to the logs. This is to be used only for local development and not in a production environment.
<b>Installed</b>	<code>"installed" =&gt; false,</code>	This flag indicates whether the ownCloud instance was installed successfully. When set to true, the install was successful. When set to false the instance is not properly installed.
<b>DB Type</b>	<code>"dbtype" =&gt; "sqlite",</code>	The dbtype indicates what type of database is being used with this install.
<b>DB Name</b>	<code>"dbname" =&gt; "owncloud",</code>	The ownCloud database within the SQL instance. ownCloud database configuration guides use "owncloud" in the instructions, it is the default for this field. This may be set to any valid table name. If the table doesn't exist, the setup will attempt to create it, provided it has proper permissions.
<b>DB User</b>	<code>"dbuser" =&gt; "",</code>	This is the user that ownCloud uses to write to the database. This must be unique across ownCloud instances using the same SQL database. The user will be created when the install wizard is complete. Since the database user in the initial login screen has admin privileges (root), ownCloud creates a new user to ensure ownCloud does not run with unnecessary database permissions. ownCloud prefixes " <b>oc_</b> " to the userid to prevent a collision between the ownCloud admin user and any existing database users. This is generated at the time of the initial login and is the Admin User as entered by the ownCloud admin.
<b>DB Password</b>	<code>"dbpassword" =&gt; "",</code>	When a new database user is created for ownCloud, a password for the database is generated by hashing the admin user password. This hash is stored in this parameter.
<b>DB Host</b>	<code>"dbhost" =&gt; "",</code>	This parameter shows the location of the database. Can be an IP pointing to the database server, a Fully Qualified Domain Name (FQDN) or localhost.
<b>DB Table Prefix</b>	<code>"dbtableprefix" =&gt; "<b>oc_</b>",</code>	All tables in the database will begin with this prefix. By default the prefix is " <b>oc_</b> ". If this prefix is changed in the config file, the tables in the database will need to be manually renamed. ownCloud will not detect a change to this option so changing this prefix without manually renaming the tables will break the instance of ownCloud.
<b>Password Salt</b>	<code>"passwordsalt" =&gt; "",</code>	This is the salt used to hash passwords within the ownCloud instance. It is important not to remove or lose this password. If lost, all passwords within the ownCloud instance will be lost.
<b>Data Directory</b>	<code>"datadirectory" =&gt; "",</code>	This parameter tells ownCloud where the default data directory is located. It is configured in the installation wizard. If using SQLITE, the database is stored in this data directory as well.
<b>Trusted Domains</b>	<code>'trusted_domains' =&gt; array ( 0 =&gt; 'A.B.C.D ', ),</code>	This parameter identifies trusted URL for ownCloud which users may log into. The initial entry '0' is automatic. If it is desired to have additional entries, these may be added manually. This is used to mitigate Header Post Poisoning attacks.

## 6.2 Reverse Proxy Configurations

The following parameters are used in the instance that Proxies are being used within the network.

Parameter	Format	Description
<b>Over-write Host</b>	“overwritehost” => “”,	By default, ownCloud attempts to detect what outside host can access the instance (www.example.com) for generating URLs. However, due to some reverse proxies, the automatically detected value may be incorrect (www.example.com:88) which would lead to incorrect URLs being generated. Use this field to enter the proper URL. If set as follows: “overwritehost” => “http://www.example.com:88”, When logging into ownCloud, the browser will point to port 88.
<b>Over-write Protocol</b>	“overwriteprotocol” => “”,	When generating URLs, ownCloud attempts to detect whether the server is accessed via https or http. However, if ownCloud is behind a proxy and the proxy handles the https calls, thereby leaving ownCloud running without SSL, ownCloud would not realize that ssl is in use which would result in incorrect URLs being generated. Valid values are “http” and “https”. If set as follows: “overwriteprotocol” => “https”, ownCloud will generate all URLs as HTTPS rather than HTTP.
<b>Over-write Web Root</b>	“overwritewebroot” => “”,	As with the host and protocol, ownCloud attempts to detect the webroot for generating URLs automatically. The webroot is the path used to access ownCloud relative to the domain, for instance, if <a href="http://www.example.com/owncloud">www.example.com/owncloud</a> is the URL pointing to the ownCloud instance, the webroot would be /owncloud. When proxies are in use, it may be difficult for ownCloud to detect this parameter resulting in invalid URLs.
<b>Over-write Cond Addr</b>	“overwritecondaddr” => “”,	
<b>Proxy</b>	“proxy” => “”,	In the instance where a proxy is required to access the internet, the proxy should be configured in this parameter. ownCloud requires access to the internet for several functions, and thus needs to have the proxy information configured to access the internet.
<b>Proxy User Password</b>	“proxyuserpwd” => “username:password”,	In the event that a proxy is configured and requires authentication, the username and password would be configured in this parameter.

## 6.3 User Experience

The following parameters are those that influence the end user’s experience.

Parameter	Format	Description
<b>Default Language</b>	“default_language” => “en”,	This is the default language for the ownCloud WebUI. When configured, the default language will be the same for all users. Users may then configure their own language preference in their Personal page. When not configured, the default language is determined from the headers sent by the web browser. For instance, if the browser is in Spanish, ownCloud will be presented in Spanish
<b>Default App</b>	“defaultapp” => “files”,	By default, when a user logs into ownCloud, they are brought to the files page. If, for instance, the admin desires a different page to be loaded upon login, configure that app here. Valid values are app id’s (for example news, files, gallery).
<b>Knowledge Base ** **Enabled</b>	“knowledgebaseenabled” => true,	When enabled, default, the help menu brings up the user documentation.
<b>Enable Avatars</b>	‘enable_avatars’ => true,	Allows for the ability to use avatars.
<b>Display Name</b>	‘allow_user_to_change_display_name’ => true,	Users can modify their display name in the Personal page. If this parameter is set to false, they may not change their display name.

## 6.4 Mail Parameters

These parameters are related to ownCloud’s ability to send emails for lost passwords or file shares.



Parameter	Format	Description
<b>Mail Domain</b>	“mail_domain” => “example.com”,	The domain to use when ownCloud sends emails. Emails can be sent in such instances as to share a public link, share notification or lost password.
<b>Mail SMTP Debug</b>	“mail_smtpdebug” => false,	
<b>Mail SMTP Mode</b>	“mail_smtpmode” => “sendmail”,	The method used to send mail. Valid values are smtp, php, sendmail and qmail. If using local or remote SMTP, set to smtp. If using PHP mail it is necessary to have an installed and working email system on the server. The program used to send email is defined in the PHP.ini file. If using sendmail, it is necessary to have an installed and working email system on the server. The sendmail binary is /usr/sbin/sendmail and should be installed on your *nix system. If using qmail to send email, the binary is /var/qmail/bin/sendmail and should be installed in your *nix system.
<b>Mail SMTP Host</b>	“mail_smtphost” => “127.0.0.1”,	Mail server host. May contain multiple hosts separated by a semi colon. Also possible is to set the port used a particular host by following the host with a colon then the port number.
<b>Mail SMTP Port</b>	“mail_smtpport” => 25,	Port used to communicate with the mail server.
<b>Mail SMTP TIME-OUT</b>	“mail_smtptimeout” => 10,	In the event that a malware or SPAM scanner is running on the SMTP server, it could be necessary to increase the SMTP timeout. That can be done using this parameter.
<b>Mail SMTP **Secure</b>	“mail_smtpsecure” => “”,	Default value is no security. May be ssl or tls depending on the required level of security.
<b>MAIL SMTP AUTH</b>	“mail_smtpauth” => false,	Determine if the mail server requires authentication. Default is false. If true, the following parameters should be configured as well.
<b>Mail SMTP Auth Type</b>	“mail_smtpauthtype” => “LOGIN”,	If SMTP authentication is required, choose the authentication type as login (default) or plain.
<b>Mail SMTP Name</b>	“mail_smtpname” => “username”,	The username to use when authentication is enabled.
<b>Mail SMTP Password</b>	“mail_smtppassword” => “password”,	The password to use when authentication is enabled.

## 6.5 Deleted Items

These parameters are related to the deleted files app.

Parameter	Format	Description
<b>Trash Bin Retention</b>	'trash-bin_retention_obligation' => 30,	When the delete app is enabled (default), this is the amount of days a file will be kept in the trash bin. Default is 30 days.
<b>Trash Bin Auto Expire</b>	'trash-bin_auto_expire' => true,	

## 6.6 Verification

This section describes different verification checks that ownCloud may perform.

Parameter	Format	Description
<b>Update Checker</b>	"updatechecker" => true,	Provides information as to whether there is a new release of ownCloud available. When enabled, default, a banner will appear on the admin's web interface when a newer version of ownCloud exists.
<b>Has Internet Connection</b>	"has_internet_connection" => true,	Alerts ownCloud if there is an internet connection (true – default). If set to false, ownCloud will not be able to look for updates, display the knowledgebase, or bring up the appstore.
<b>Working WebDAV</b>	"check_for_working_webdav" => true,	Alerts ownCloud to verify a working WebDAV connection. This is done by attempting to make a WebDAV request from PHP.
<b>Working .htaccess</b>	"check_for_working_htaccess" => true,	Verifies whether the .htaccess file may be modified by ownCloud. If set to false, this check will not be performed. If the file cannot be modified, items such as large file uploads cannot be performed. This check only affects Apache servers.

## 6.7 Logging

This section describes parameters associated with ownCloud's logging abilities.

Parameter	Format	Description
<b>Log Type</b>	“log_type” => “owncloud”,	By default the ownCloud logs are sent to the owncloud.log file within the default data directory. If syslogging is desired, set this parameter to syslog.
<b>Log File</b>	“logfile” => “”,	The log file, by default, is owncloud.log and stored in the default data directory. Use this parameter to change the name to something other than owncloud.log.
<b>Log Level</b>	“loglevel” => “”,	ownCloud has several levels of logging. This may be set on the Admin page of the webUI or directly in the configuration file using this parameter. Valid values are: 0=Debug, 1=Info, 2=Warning, 3=Error. The default value is Warning
<b>Log Date Format</b>	‘logdateformat’ => ‘F d, Y H:i:s’,	ownCloud allows the admin to specify the format of the time and date within the log file. Valid values may be found at the following website: <a href="http://www.php.net/manual/en/function.date.php">http://www.php.net/manual/en/function.date.php</a> .
<b>Log Time Zone</b>	‘logtimezone’ => ‘Europe/Berlin’,	By default, the time zone displayed in the ownCloud logs is UTC. To change the displayed time zone to the local time zone, use this parameter. For a list of valid values, see the following website: <a href="http://php.net/manual/en/timezones.php">http://php.net/manual/en/timezones.php</a> .
<b>Log Query</b>	“log_query” => false,	When set to “true”, all SQL queries performed by ownCloud will be written to the log file. Default is false. It is not recommended to run with this enabled as it will fill up the log file. Use only for debugging purposes.
<b>Log Auth Fail IP</b>	“log_authfailip” => false,	When set to true, the IP addresses of failed login attempts will be logged.
<b>Log Rotate Size</b>	‘log_rotate_size’ => false, // 104857600, // 100 MiB	Since ownCloud log files can get large in size, this parameter may be used to rotate to a new log file once it reaches the specified size. This should be configured in bytes. Default is false, or 0, which will not rotate the file.

## 6.8 Session Info

The following parameters are related to sessions within ownCloud.

Parameter	Format	Description
<b>Remember Cookie Lifetime</b>	“remember_login_cookie_lifetime” => 60*60*24*15,	ownCloud provides the user the option of remembering their login credentials (this option appears as the “remember” checkbox on the login screen). This parameter allows the admin to configure the length of time which ownCloud will remember that user. Default is 15 days. The configuration is in seconds.
<b>Session Lifetime</b>	“session_lifetime” => 60 * 60 * 24,	ownCloud will automatically logout a user after a period of inactivity. The default is 1 day. This parameter can be used to modify that time. Configuration is in seconds.

## 6.9 Code Locations

ownCloud has the ability to find parts of its code in non-standard locations. This section describes how to configure ownCloud for that functionality.

Parameter	Format	Description
<b>Theme</b>	<code>"theme" =&gt; "",</code>	If the instance of ownCloud is themed, the name of the theme should be configured here. For more information on this parameter, see the document on Theming.
<b>3rd Party Root</b>	<code>"3rdpartyroot" =&gt; "",</code>	ownCloud uses some 3rd party PHP components to provide certain functionality. These components are shipped as part of the software package and reside in <code>~/owncloud/3rdparty</code> . However, if this folder resides elsewhere, the location can be configured here. For example <code>/srv/http/path/to/3rdparty</code> .
<b>3rd Party URL</b>	<code>"3rdpartyurl" =&gt; "",</code>	In the event that the 3rdpartyroot is configured, this parameter should be configured as well to show the http web path to the 3rdpartyroot starting at the owncloud web root. For instance <code>/path/to/3rdparty</code> .
<b>Custom Client</b>	<code>'customclient_desktop' =&gt; ' //http://owncloud.org/sync-clients/ 'customclient_android' =&gt; ' //https://play.google.com/store/apps/details?id=com.owncloud.android 'customclient_ios' =&gt; ' //https://itunes.apple.com/us/app/owncloud/id543672169?mt=8</code>	The location where ownCloud will bring the user to download clients. The link is in the first run wizard or the Personal page.

## 6.10 APPS

The following parameters are used for ownCloud apps.

Parameter	Format	Description
<b>Apps Paths</b>	<pre> “apps_paths” =&gt; array( 0 =&gt; array( ‘path’ =&gt; ‘/var/www/owncloud/apps’, ‘url’ =&gt; ‘/apps’, ‘writable’ =&gt; true, ), 1 =&gt; array ( ‘path’ =&gt; ‘/var/www/owncloud/apps2’, ‘url’ =&gt; ‘/apps2’, Writable =&gt; false, ), ), </pre>	Use this parameter to set the location of the apps folder which should be scanned for available apps and/or where user specific apps should be installed. The 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.
<b>App Store Enabled</b>	<pre> “appstoreenabled” =&gt; true, </pre>	When enabled, admins may install apps from the ownCloud app store.
<b>App Store URL</b>	<pre> “appstoreurl” =&gt; “http://api.apps.owncloud.com/v1”, </pre>	The URL of the appstore.
<b>App Code Checker</b>	<pre> “appcodechecker” =&gt; “”, </pre>	Checks for malicious code fragments of 3 rd party apps.

## 6.11 Previews

ownCloud allows for thumbnail previews of files. This section contains the different configuration parameters available for that functionality.

Parameter	Format	Description
<b>Enable Previews</b>	<pre> ‘enable_previews’ =&gt; true, </pre>	When enabled, default, the user will have file thumbnails visible. Disable to remove thumbnails.
<b>Preview Width</b>	<pre> ‘preview_max_x’ =&gt; null, </pre>	Maximum width of the thumbnail. Default is null meaning no limit.
<b>Preview Height</b>	<pre> ‘preview_max_y’ =&gt; null, </pre>	The maximum height of the thumbnail. Default is set to null meaning no limit.
<b>Scale Factor</b>	<pre> ‘preview_max_scale_factor’ =&gt; 10, </pre>	Scale the thumbnail by this factor. Default is 10.
<b>Libreoffice Path</b>	<pre> ‘preview_libreoffice_path’ =&gt; ‘/usr/bin/libreoffice’, </pre>	ownCloud uses Libre Office for previews. This parameter indicates the location of the Libre Office executable.
<b>Libreoffice Parameters</b>	<pre> ‘pre- view_office_cl_parameters’ =&gt; “”, </pre>	Use this if Libre Office requires additional arguments

## 6.12 Maintenance

This section discusses the different stages of maintenance.

Parameter	Format	Description
<b>Single User</b>	<code>'singleuser' =&gt; false,</code>	When set to true, the ownCloud instance will be unavailable for all users not in the admin group. This is useful when performing maintenance.
<b>Maintenance</b>	<code>"maintenance" =&gt; false,</code>	Enable maintenance mode to disable ownCloud. When performing upgrades, ownCloud automatically enters maintenance mode. When enabled, users who are already logged-in are kicked out of ownCloud instantly.

## 6.13 Miscellaneous

The remaining parameters are listed below.

Parameter	Format	Description
<b>Open SSL</b>	<code>'openssl' =&gt; array( // 'config' =&gt; '/absolute/location/of/openssl.cnf', ),</code>	
<b>User Backends</b>	<code>'user_backends' =&gt; array( array( 'class' =&gt; 'OC_User_IMAP', 'arguments' =&gt; array( '{imap.gmail.com:993/imap/ssl}INBOX', 'user' =&gt; 'user', 'password' =&gt; 'password' ), ), ),</code>	It is possible to configure additional user backends in ownCloud. The “External user Support” (user_external) app provides the following backends: IMAP (OC_User_IMAP), SMB (OC_User_SMB), FTP (OC_User_FTP).
<b>CSP Policy</b>	<code>"custom_csp_policy" =&gt; "default-src 'self'; script-src 'self' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; frame-src *; img-src *; font-src 'self' data:; media-src *",</code>	
<b>X Frame</b>	<code>"xframe_restriction" =&gt; true,</code>	XFrame-restriction is a header which prevents browsers from showing the site inside an iframe. This may be used to prevent clickjacking.
<b>Mem Cache</b>	<code>"memcached_server" =&gt; array('localhost', '11211'),</code>	Server details for one or more memcached servers to use for memory caching. Memcache is only used if other memory cache options (xcache, apc, apcu) are not available.
<b>Force SSL</b>	<code>"forcessl" =&gt; false,</code>	If the admin checks “Enforce HTTPS” in the Admin page of the ownCloud WebUI. This will be set to true indicating that only HTTPS may be used to access this instance of ownCloud. HTTP requests will be denied.
<b>Black List</b>	<code>"blacklisted_files" =&gt; array('.htaccess'),</code>	Files listed in this array will not be uploaded to ownCloud. It should be noted that wildcards are not supported in this array. The configured must be exact file names to be blocked. If wildcards are required, use the ownCloud firewall.

ownCloud uses a configuration file to set certain parameters. The configuration file is entitled *config.php* and resides in the *config* directory of the ownCloud installation. Also residing in that directory is a sample configuration file entitled *config.sample.php*. This file lists all the configurable parameters within ownCloud along with a brief description. This document will provide more details as to what each parameter is used for.

## 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 [here](#) report it to the apps repository

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





## INDICES AND TABLES

- *genindex*