

ownCloud Server Administration Manual Release 9.0

The ownCloud developers

January 24, 2018

CONTENTS

1	ownCloud 9.0 Server Administration Manual Introduction11.1Introduction11.2ownCloud Videos and Blogs11.3Target Audience1
2	ownCloud 9.0 Release Notes 3 2.1 Changes in 9.0.11 3 2.2 Changes in 9.0 3 2.3 Enterprise 9.0 4 2.4 Changes in 8.2 4 2.5 Changes in 8.1 4 2.6 Enterprise 8.1 Only 5 2.7 ownCloud 8.0 5 2.8 Enterprise 8.0 Only 7 2.9 ownCloud 7 Release Notes 7 2.10 Enterprise 7 Only 9
3	What's New for Admins in ownCloud 9.0113.1Enterprise Only11
4	Installation134.1System Requirements134.2ownCloud Deployment Recommendations154.3Preferred Linux Installation Method224.4Installation Wizard254.5Installing ownCloud From the Command Line304.6Changing the Web Route314.7Installing and Managing Apps324.8Supported Apps in ownCloud344.9Manual Installation on Linux354.10ownCloud Community Appliance424.11Installing PHP 5.4 on RHEL 6 and CentOS 6444.12Installing PHP 5.5 on RHEL 7 and CentOS 7454.14nginx Example Configuration49
5	ownCloud Server Configuration595.1Warnings on Admin Page595.2Importing System-wide and Personal SSL Certificates615.3Using the occ Command625.4Configuring the Activity App82

	5.5	Configuring the ClamAV Antivirus Scanner
	5.6	Configuring Memory Caching 85
	5.7	Background Jobs
	5.8	Config.php Parameters
	5.9	Email Configuration
	5.10	Linking External Sites
	5.11	Custom Client Download Repositories
	5.12	Knowledge Base Configuration 122
	5.13	Language Configuration 122
	5.14	Logging Configuration 123
	5.15	Hardening and Security Guidance
	5.16	Reverse Provy Configuration 127
	5.17	Using Third Party PHP Components
	5.19	InveScript and CSS Asset Management
	5.10	Javascript and CSS Asset Management
	5.19	
	5.20	
	5.21	Enable index.pnp-less URLs
6	Lleor	Managamant 135
U	C 1	Ulan Management 125
	0.1	User Management
	0.2	
	0.3	
	6.4	User Authentication with IMAP, SMB, and FTP
	6.5	User Authentication with LDAP
	6.6	LDAP User Cleanup
	6.7	User Provisioning API
7	Eile 6	Shaving and Managament 172
/	rne S	Tring and Wanagement 173
	7.1	File Snanng
	1.2	
	1.3	Uploading big files > 512 MB
	1.4	Configuring the Collaborative Documents App
	7.5	Providing Default Files
	7.6	Configuring External Storage (GUI)
	7.7	Configuring External Storage (Configuration File)
	7.8	External Storage Authentication mechanisms
	7.9	Encryption Configuration
	7.10	Transactional File Locking
	7.11	Previews Configuration
	7.12	Controlling File Versions and Aging
	_	
8	Data	base Configuration 221
	8.1	Converting Database Type
	8.2	Database Configuration
0		
9	Mim	etypes Management 229
	9.1	Mimetype Aliases
	9.2	Mimetype mapping
	9.3	Icon retrieval
10		
10	Main	tenance 231
	10.1	Maintenance Mode Configuration
	10.2	Backing up ownCloud
	10.3	How to Upgrade Your ownCloud Server
	10.4	Upgrade ownCloud From Packages

	10.5	Upgrading ownCloud with the Updater App	237
	10.0	Manual ownCloud Opgrade	240
	10.7	Restoring ownCloud	241
	10.8	Migrating to a Different Server	242
11	Issue	s and Troubleshooting	245
	11.1	General Troubleshooting	245
	11.2	Code Signing	250
12	Enter	prise Edition Only	255
	12.1	Enterprise Edition Installation	255
	12.2	Creating Branded ownCloud Clients (Enterprise only)	261
	12.3	Enterprise Server Branding (Enterprise only)	262
	12.4	External Storage (Enterprise only)	264
	12.5	User Management (Enterprise only)	284
	12.6	Enterprise File Management (Enterprise Only)	293
	12.7	Enterprise Logging Apps (Enterprise only)	300
	12.8	Enterprise Firewall (Enterprise only)	301
	12.9	Enterprise Troubleshooting	305

CHAPTER

ONE

OWNCLOUD 9.0 SERVER ADMINISTRATION MANUAL INTRODUCTION

1.1 Introduction

Welcome to the ownCloud Server Administration Guide. This guide describes administration tasks for ownCloud, the flexible open source file synchronization and sharing solution. ownCloud includes the ownCloud server, which runs on Linux, client applications for Microsoft Windows, Mac OS X and Linux, and mobile clients for the Android and Apple iOS operating systems.

Current editions of ownCloud manuals are always available online at doc.owncloud.org and doc.owncloud.com.

ownCloud server is available in three editions:

- The free community-supported Server. This is the core server for all editions.
- The Standard Subscription for customers who want paid support for the core Server, without Enterprise applications.
- The Enterprise Subscription provides paid support for the Enterprise Edition. This includes the core Server and Enterprise apps.

See What's New for Admins in ownCloud 9.0 for more information on the different ownCloud editions.

1.2 ownCloud Videos and Blogs

See the official ownCloud channel and ownClouders community channel on YouTube for tutorials, overviews, and conference videos.

Visit ownCloud Planet for news and developer blogs.

1.3 Target Audience

This guide is for users who want to install, administer, and optimize their ownCloud servers. To learn more about the ownCloud Web user interface, and desktop and mobile clients, please refer to their respective manuals:

- ownCloud User Manual
- ownCloud Desktop Client
- ownCloud Android App
- ownCloud iOS App

CHAPTER

OWNCLOUD 9.0 RELEASE NOTES

2.1 Changes in 9.0.11

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 9.0.11 that need your attention:

The full ownCloud Server 9.0.11 changelog can be found here: https://owncloud.org/changelog/#latest9.0

Added "occ files:scan" repair mode to repair filecache inconsistencies We recommend to use this command when directed to do so in the upgrade process. Please check the ownCloud documentation for more information (https://doc.owncloud.com/server/10.0/admin_manual/configuration/server/occ_command.html?highlight=occ#the-repair-option).

2.2 Changes in 9.0

9.0 requires .ico files for favicons. This will change in 9.1, which will use .svg files. See Changing favicon in the Developer Manual.

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

The Calendar and Contacts apps have been rewritten and the CalDAV and CardDAV backends of these apps were merged into ownCloud core. During the upgrade existing Calendars and Addressbooks are automatically migrated (except when using the IMAP user backend). As a fallback for failed upgrades, when using the IMAP user backend or as an option to test a migration dav:migrate-calendars and/or dav:migrate-addressbooks scripts are available (only in oC 9.0) via the occ command. See Using the occ Command.

Warning: After upgrading to ownCloud 9.0 and **before** continuing to upgrade to 9.1 make sure that all of your and your users Calendars and Addressbooks are migrated correctly. Especially when using the IMAP user backend (other user backends might be also affected) you need to manually run the mentioned occ migration commands described above.

Updates on systems with large datasets will take longer, due to the addition of checksums to the oC database. See https://github.com/owncloud/core/issues/22747.

Linux packages are available from our official download repository. New in 9.0: split packages. owncloud installs ownCloud plus dependencies, including Apache and PHP. owncloud-files installs only ownCloud. This is useful for custom LAMP stacks, and allows you to install your own LAMP apps and versions without packaging conflicts with ownCloud. See *Preferred Linux Installation Method*.

New option for the ownCloud admin to enable or disable sharing on individual external mountpoints (see *Mount Options*). Sharing on such mountpoints is disabled by default.

2.3 Enterprise 9.0

owncloud-enterprise packages are no longer available for CentOS6, RHEL6, Debian7, or any version of Fedora. A new package, owncloud-enterprise-files, is available for all supported platforms. This new package comes without dependencies, and is installable on a larger number of platforms. System administrators on these older distros must install their own LAMP stacks and databases. On newer supported distros, install owncloud-enterprise as usual. See *Installing & Upgrading ownCloud Enterprise Edition*.

2.4 Changes in 8.2

New location for Linux package repositories; ownCloud admins must manually change to the new repos. See *How to* Upgrade Your ownCloud Server

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

filesystem_check_changes in config.php is set to 0 by default. This prevents unnecessary update checks and improves performance. If you are using external storage mounts such as NFS on a remote storage server, set this to 1 so that ownCloud will detect remote file changes.

XSendFile support has been removed, so there is no longer support for serving static files from your ownCloud server.

LDAP issue: 8.2 uses the memberof attribute by default. If this is not activated on your LDAP server your user groups will not be detected, and you will see this message in your ownCloud log: Error PHP Array to string conversion at /var/www/html/owncloud/lib/private/template/functions.php#36. Fix this by disabling the memberof attribute on your ownCloud server with the occ command, like this example on Ubuntu Linux:

sudo -u www-data php occ ldap:set-config "s01" useMemberOfToDetectMembership 0

Run sudo -u www-data php occ ldap:show-config to find the correct sNN value; if there is not one then use empty quotes, "". (See *Using the occ Command*.)

Users of the Linux Package need to update their repository setup as described in this blogpost.

2.5 Changes in 8.1

Use APCu only if available in version 4.0.6 and higher. If you install an older version, you will see a APCu below version 4.0.6 is installed, for stability and performance reasons we recommend to update to a newer APCu version warning on your ownCloud admin page.

SMB external storage now based on php5-libsmbclient, which must be downloaded from the ownCloud software repositories (installation instructions).

"Download from link" feature has been removed.

The .htaccess and index.html files in the data/ directory are now updated after every update. If you make any modifications to these files they will be lost after updates.

The SabreDAV browser at /remote.php/webdav has been removed.

Using ownCloud without a trusted_domain configuration will not work anymore.

The logging format for failed logins has changed and considers now the proxy configuration in config.php.

A default set of security and privacy HTTP headers have been added to the ownCloud .htaccess file, and ownCloud administrators may now customize which headers are sent.

More strict SSL certificate checking improves security but can result in "cURL error 60: SSL certificate problem: unable to get local issuer certificate" errors with certain broken PHP versions. Please verify your SSL setup, update your PHP or contact your vendor if you receive these errors.

The persistent file-based cache (e.g. used by LDAP integration) has been dropped and replaced with a memory-only cache, which must be explicitly configured. See *User Authentication with LDAP*. Memory cache configuration for the ownCloud server is no longer automatic, requiring installation of your desired cache backend and configuration in config.php (see *Configuring Memory Caching*.)

The OC_User_HTTP backend has been removed. Administrators are encouraged to use the user_webdavauth application instead.

ownCloud ships now with its own root certificate bundle derived from Mozilla's root certificates file. The system root certificate bundle will not be used anymore for most requests.

When you upgrade from ownCloud 8.0, with encryption enabled, to 8.1, you must enable the new encryption backend and migrate your encryption keys. See *Encryption migration to ownCloud* 8.0.

Encryption can no longer be disabled in ownCloud 8.1. It is planned to re-add this feature to the command line client for a future release.

It is not recommended to upgrade encryption-enabled systems from ownCloud Server 8.0 to version 8.1.0 as there is a chance the migration will break. We recommend migrating to the first bugfix release, ownCloud Server 8.1.1.

Due to various technical issues, by default desktop sync clients older than 1.7 are not allowed to connect and sync with the ownCloud server. This is configurable via the minimum.supported.desktop.version switch in config.php.

Previews are now generated at a maximum size of 2048 x 2048 pixels. This is configurable via the preview_max_x and preview_max_y switches in config.php.

The ownCloud 8 server is not supported on any version of Windows.

The 8.1.0 release has a minor bug which makes app updates fail at first try. Reload the apps page and try again, and the update will succeed.

The forcessl option within the config.php and the Enforce SSL option within the Admin-Backend was removed. This now needs to be configured like described in *Use HTTPS*.

WebDAV file locking was removed in oC 8.1 which causes Finder on Mac OS X to mount WebDAV read-only.

2.6 Enterprise 8.1 Only

The SharePoint Drive app does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

2.7 ownCloud 8.0

2.7.1 Manual LDAP Port Configuration

When you are configuring the LDAP user and group backend application, ownCloud may not auto-detect the LDAP server's port number, so you will need to enter it manually.

2.7.2 No Preview Icon on Text Files

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

2.7.3 Remote Federated Cloud Share Cannot be Reshared With Local Users

When you mount a Federated Cloud share from a remote ownCloud server, you cannot re-share it with your local ownCloud users. (See *Configuring Federation Sharing* to learn more about federated cloud sharing)

2.7.4 Manually Migrate Encryption Keys after Upgrade

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

2.7.5 Windows Server Not Supported

Windows Server is not supported in ownCloud 8.

2.7.6 PHP 5.3 Support Dropped

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

2.7.7 Disable Apache Multiviews

If Multiviews are enabled in your Apache configuration, this may cause problems with content negotiation, so disable Multiviews by removing it from your Apache configuration. Look for lines like this:

<Directory /var/www/owncloud>
Options Indexes FollowSymLinks Multiviews

Delete Multiviews and restart Apache.

2.7.8 ownCloud Does Not Follow Symlinks

ownCloud's file scanner does not follow symlinks, which could lead to infinite loops. To avoid this do not use soft or hard links in your ownCloud data directory.

2.7.9 No Commas in Group Names

Creating an ownCloud group with a comma in the group name causes ownCloud to treat the group as two groups.

2.7.10 Hebrew File Names Too Large on Windows

On Windows servers Hebrew file names grow to five times their original size after being translated to Unicode.

2.7.11 Google Drive Large Files Fail with 500 Error

Google Drive tries to download the entire file into memory, then write it to a temp file, and then stream it to the client, so very large file downloads from Google Drive may fail with a 500 internal server error.

2.7.12 Encrypting Large Numbers of Files

When you activate the Encryption app on a running server that has large numbers of files, it is possible that you will experience timeouts. It is best to activate encryption at installation, before accumulating large numbers of files on your ownCloud server.

2.8 Enterprise 8.0 Only

2.8.1 Sharepoint Drive SSL Not Verified

The SharePoint Drive app does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

2.8.2 No Federated Cloud Sharing with Shibboleth

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

2.8.3 Direct Uploads to SWIFT do not Appear in ownCloud

When files are uploaded directly to a SWIFT share mounted as external storage in ownCloud, the files do not appear in ownCloud. However, files uploaded to the SWIFT mount through ownCloud are listed correctly in both locations.

2.8.4 SWIFT Objectstore Incompatible with Encryption App

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

2.8.5 App Store is Back

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

2.9 ownCloud 7 Release Notes

2.9.1 Manual LDAP Port Configuration

When you are configuring the LDAP user and group backend application, ownCloud may not auto-detect the LDAP server's port number, so you will need to enter it manually.

2.9.2 LDAP Search Performance Improved

Prior to 7.0.4, LDAP searches were substring-based and would match search attributes if the substring occurred anywhere in the attribute value. Rather, searches are performed on beginning attributes. With 7.0.4, searches will match at the beginning of the attribute value only. This provides better performance and a better user experience.

Substring searches can still be performed by prepending the search term with "*".For example, a search for te will find Terri, but not Nate:

```
occ ldap:search "te"
```

If you want to broaden the search to include Nate, then search for *te:

```
occ ldap:search "*te"
```

Refine searches by adjusting the User Search Attributes field of the Advanced tab in your LDAP configuration on the Admin page. For example, if your search attributes are givenName and sn you can find users by first name + last name very quickly. For example, you'll find Terri Hanson by searching for te ha. Trailing whitespaces are ignored.

2.9.3 Protecting ownCloud on IIS from Data Loss

Under certain circumstances, running your ownCloud server on IIS could be at risk of data loss. To prevent this, follow these steps.

- In your ownCloud server configuration file, owncloud\config\config.php, set config_is_read_only to true.
- Set the config.php file to read-only.
- When you make server updates config.php must be made writeable. When your updates are completed re-set it to read-only.

2.9.4 Antivirus App Modes

The Antivirus App offers three modes for running the ClamAV anti-virus scanner: as a daemon on the ownCloud server, a daemon on a remote server, or an executable mode that calls clamscan on the local server. We recommend using one of the daemon modes, as they are the most reliable.

2.9.5 "Enable Only for Specific Groups" Fails

Some ownCloud applications have the option to be enabled only for certain groups. However, when you select specific groups they do not get access to the app.

2.9.6 Changes to File Previews

For security and performance reasons, file previews are available only for image files, covers of MP3 files, and text files, and have been disabled for all other filetypes. Files without previews are represented by generic icons according to their file types.

2.9.7 4GB Limit on SFTP Transfers

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

2.9.8 "Not Enough Space Available" on File Upload

Setting user quotas to unlimited on an ownCloud installation that has unreliable free disk space reporting— for example, on a shared hosting provider— may cause file uploads to fail with a "Not Enough Space Available" error. A workaround is to set file quotas for all users instead of unlimited.

2.9.9 No More Expiration Date On Local Shares

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

2.9.10 Zero Quota Not Read-Only

Setting a user's storage quota should be the equivalent of read-only, however, users can still create empty files.

2.10 Enterprise 7 Only

2.10.1 No Federated Cloud Sharing with Shibboleth

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

2.10.2 Windows Network Drive

Windows Network Drive runs only on Linux servers because it requires the Samba client, which is included in all Linux distributions.

php5-libsmbclient is also required, and there may be issues with older versions of libsmbclient; see Using External Storage > Installing and Configuring the Windows Network Drive App in the Enterprise Admin manual for more information.

By default CentOS has activated SELinux, and the httpd process can not make outgoing network connections. This will cause problems with curl, Idap and samba libraries. Again, see Using External Storage > Installing and Configuring the Windows Network Drive App in the Enterprise Admin manual for instructions.

2.10.3 Sharepoint Drive SSL

The SharePoint Drive app does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

2.10.4 Shibboleth and WebDAV Incompatible

Shibboleth and standard WebDAV are incompatible, and cannot be used together in ownCloud. If Shibboleth is enabled, the ownCloud client uses an extended WebDAV protocol

2.10.5 No SQLite

SQLite is no longer an installation option for ownCloud Enterprise Edition, as it not suitable for multiple-user installations or managing large numbers of files.

2.10.6 No App Store

The App Store is disabled for the Enterprise Edition.

2.10.7 LDAP Home Connector Linux Only

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

THREE

WHAT'S NEW FOR ADMINS IN OWNCLOUD 9.0

See the ownCloud 9.0 Features page on Github for a comprehensive list of new features and updates.

ownCloud has many improvements. Some of our new features are:

- Split Linux packaging, dividing ownCloud and dependencies into two separate packages (*Preferred Linux Installation Method*)
- Separate encryption for home storage and remote storage; you may encrypt remote storage without encrypting local storage. (*Encryption Configuration*)
- New command to transfer files from one user to another. (Transferring Files to Another User)
- Streamlined Federation sharing with user and group name auto-fill. See (*Creating a new Federation Share (9.0+ only)*)
- Configurable password reset URL. See (*Resetting a User Password*)
- Command-line options added to the Updater app. (Upgrading ownCloud with the Updater App)
- Many new occ commands. (Using the occ Command)
- Admin option to enable and disable sharing on external storage mountpoints. (Mount Options)
- New occ commands for migrating contacts and calendars from 8.2, if auto-migration during upgrade fails, and new commands for creating addressbooks and calendars (*Dav Commands*)
- New optional second name attribute in the LDAP app, so that user names appear as User Foo (optional 2nd attribute) (*Directory Settings*)

3.1 Enterprise Only

- Advanced tagging management with the Workflow app. (*Advanced File Tagging With the Workflow App (Enterprise only*))
- Advanced authentication backends. (Enterprise-Only Authentication Options)
- Password policy app for share links, for setting password requirements such as minimum length and required characters. (*Share Link Password Policy*)

CHAPTER

FOUR

INSTALLATION

4.1 System Requirements

4.1.1 Memory

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

4.1.2 Recommended Setup for Running ownCloud

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

- Ubuntu 16.04
- MySQL/MariaDB
- PHP 7.0
- Apache 2.4 with mod_php

4.1.3 Supported Platforms

If you are not able to use one or more of the above tools, the following options are also supported.

Server

- Debian 7 and 8
- SUSE Linux Enterprise Server 12 and 12 SP1
- Red Hat Enterprise Linux/Centos 6.5 and 7 (7 is 64-bit only)
- Ubuntu 14.04 LTS

Web Server

• Apache 2.4 with mod_php

Databases

- Oracle 11g (Enterprise edition only)
- PostgreSQL

Hypervisors

- Hyper-V
- VMware ESX
- Xen
- KVM

Desktop

- Windows 7+
- Mac OS X 10.7+ (64-bit only)
- Ubuntu 16.10
- Ubuntu 16.04
- Ubuntu 14.04
- Debian 7.0
- Debian 8.0
- CentOS 7
- Fedora 24
- Fedora 25
- openSUSE Leap 42.1
- openSUSE Leap 42.2

Note: For Linux distributions, we support, if technically feasible, the latest 2 versions per platform and the previous LTS.

Mobile

- iOS 9.0+
- Android 4.0+

Web Browser

- IE11+ (except Compatibility Mode)
- Firefox 14+
- Chrome 18+
- Safari 5+

See Manual Installation on Linux for minimum software versions for installing ownCloud.

4.1.4 Database Requirements for MySQL / MariaDB

The following is currently required if you're running ownCloud together with a MySQL / MariaDB database:

- Disabled or BINLOG_FORMAT = MIXED configured Binary Logging (See: *MySQL / MariaDB with Binary Logging Enabled*)
- InnoDB storage engine (MyISAM is not supported, see: MySQL / MariaDB storage engine)
- "READ COMMITED" transaction isolation level (See: *MySQL / MariaDB* "*READ COMMITED*" transaction isolation level)

4.2 ownCloud Deployment Recommendations

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

4.2.1 General Recommendations

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

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

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

4.2.2 Small Workgroups or Departments

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

Recommended System Requirements

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

Authentication via an existing LDAP or Active Directory server.

• Components One server with at least 2 CPU cores, 16GB RAM, local storage as needed.



- **Operating system** Enterprise-grade Linux distribution with full support from OS vendor. We recommend Red Hat Enterprise Linux or SUSE Linux Enterprise Server 12.
- SSL Configuration The SSL termination is done in Apache. A standard SSL certificate is needed, installed according to the Apache documentation.
- Load Balancer None.
- Database MySQL, MariaDB or PostgreSQL. We currently recommend MySQL / MariaDB, as our customers have had good experiences when moving to a Galera cluster to scale the DB.
- **Backup** Install owncloud, ownCloud data directory and database on Btrfs filesystem. Make regular snapshots at desired intervals for zero downtime backups. Mount DB partitions with the "nodatacow" option to prevent fragmentation.

Alternatively, make nightly backups with service interruption:

- Shut down Apache.
- Create database dump.
- Push data directory to backup.
- Push database dump to backup.
- Start Apache.

Then optionally rsync to a backup storage or tape backup. (See the Maintenance section of the Administration manual for tips on backups and restores.)

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

4.2.3 Mid-sized Enterprises

- Number of users 150 to 1,000 users.
- Storage size Up to 200TB.

• **High availability level** Every component is fully redundant and can fail without service interruption. Backups without service interruption

Recommended System Requirements

2 to 4 application servers.

A cluster of two database servers.

Storage on an NFS server.

Authentication via an existing LDAP or Active Directory server.



• Components

- 2 to 4 application servers with 4 sockets and 32GB RAM.
- 2 DB servers with 4 sockets and 64GB RAM.
- 1 HAproxy load balancer with 2 sockets and 16GB RAM.
- NFS storage server as needed.
- **Operating system** Enterprise grade Linux distribution with full support from OS vendor. Red Hat Enterprise Linux or SUSE Linux Enterprise Server 12 are recommended.
- SSL Configuration The SSL termination is done in the HAProxy load balancer. A standard SSL certificate is needed, installed according to the HAProxy documentation.
- Load Balancer HAProxy running on a dedicated server in front of the application servers. Sticky session needs to be used because of local session management on the application servers.
- Database MySQL/MariaDB Galera cluster with master-master replication.
- **Backup** Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.
 - Create a snapshot on the NFS storage server.
 - At the same time stop the MySQL replication.
 - Create a MySQL dump of the backup slave.
 - Push the NFS snapshot to the backup.
 - Push the MySQL dump to the backup.
 - Delete the NFS snapshot.

- Restart MySQL replication.
- Authentication User authentication via one or several LDAP or Active Directory servers. (See User Authentication with LDAP for information on configuring ownCloud to use LDAP and AD.)
- LDAP Read-only slaves should be deployed on every application server for optimal scalability
- Session Management Session management on the application server. PHP sessions are stored in a tmpfs mounted at the operating system-specific session storage location. You can find out where that is by running grep -R 'session.save_path' /etc/php5 and then add it to the /etc/fstab file, for example: echo "tmpfs /var/lib/php5/pool-www tmpfs defaults, noatime, mode=1777 0 0" >> /etc/fstab.
- **Memory Caching** A memcache speeds up server performance, and ownCloud supports four memcaches; refer to Configuring Memory Caching for information on selecting and configuring a memcache.
- Storage Use an off-the-shelf NFS solution, such as IBM Elastic Storage or RedHat Ceph.
- **ownCloud Edition** Enterprise Edition. (See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.)

4.2.4 Large Enterprises and Service Providers

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

Recommended System Requirements

4 to 20 application/Web servers.

A cluster of two or more database servers.

Storage is an NFS server, or an object store that is S3 compatible.

Cloud federation for a distributed setup over several data centers.

Authentication via an existing LDAP or Active Directory server, or SAML.

- Components
 - 4 to 20 application servers with 4 sockets and 64GB RAM.
 - 4 DB servers with 4 sockets and 128GB RAM
 - 2 Hardware load balancer, for example BIG IP from F5
 - NFS storage server as needed.
- **Operating system** RHEL 7 with latest service packs.
- SSL Configuration The SSL termination is done in the load balancer. A standard SSL certificate is needed, installed according to the load balancer documentation.
- Load Balancer A redundant hardware load-balancer with heartbeat, for example F5 Big-IP. This runs two load balancers in front of the application servers.
- Database MySQL/MariaDB Galera Cluster with 4x master master replication.



- **Backup** Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.
 - Create a snapshot on the NFS storage server.
 - At the same time stop the MySQL replication.
 - Create a MySQL dump of the backup slave.
 - Push the NFS snapshot to the backup.
 - Push the MySQL dump to the backup.
 - Delete the NFS snapshot.
 - Restart MySQL replication.
- Authentication User authentication via one or several LDAP or Active Directory servers, or SAML/Shibboleth. (See User Authentication with LDAP and Shibboleth Integration.)
- LDAP Read-only slaves should be deployed on every application server for optimal scalability.
- Session Management Redis should be used for the session management storage.
- Caching Redis for distributed in-memory caching (see Configuring Memory Caching).
- **Storage** An off-the-shelf NFS solution should be used. Examples are IBM Elastic Storage or RedHAT Ceph. Optionally, an S3 compatible object store can also be used.
- **ownCloud Edition** Enterprise Edition. (See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.)

4.2.5 Hardware Considerations

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

Single Machine / Scale-Up Deployment

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

Pros:

- Easy setup: no session storage daemon, use tmpfs and memory caching to enhance performance, local storage.
- No network latency to consider.
- To scale buy a bigger CPU, more memory, larger hard drive, or additional hard drives.

Cons:

- Fewer high availability options.
- The amount of data in ownCloud tends to continually grow. Eventually a single machine will not scale; I/O performance decreases and becomes a bottleneck with multiple up- and downloads, even with solid-state drives.

Scale-Out Deployment

Provider setup:

- DNS round robin to HAProxy servers (2-n, SSL offloading, cache static resources)
- Least load to Apache servers (2-n)
- Memcached/Redis for shared session storage (2-n)
- Database cluster with single Master, multiple slaves and proxy to split requests accordingly (2-n)
- GPFS or Ceph via phprados (2-n, 3 to be safe, Ceph 10+ nodes to see speed benefits under load)

Pros:

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

Cons:

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

What About Nginx / PHP-FPM?

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

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

When master fails another slave can become master. However, the increased complexity carries some risks: Multimaster has the risk of split brain, and deadlocks. ownCloud tries to solve the problem of deadlocks with high-level file locking.

4.2.6 Software Considerations

Operating System

We are dependent on distributions that offer an easy way to install the various components in up-to-date versions. ownCloud has a partnership with RedHat and SUSE for customers who need commercial support. Canonical, the parent company of Ubuntu Linux, also offers enterprise service and support. Debian and Ubuntu are free of cost, and include newer software packages. CentOS is the community-supported free-of-cost Red Hat Enterprise Linux clone. openSUSE is community-supported, and includes many of the same system administration tools as SUSE Linux Enterprise Server.

Web server

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

Relational Database

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

The second best option is PostgreSQL (alter table does not lock table, which makes migration less painful) although we have yet to find a customer who uses a master-slave setup.

What about the other DBMS?

- Sqlite is adequate for simple testing, and for low-load single-user deployments. It is not adequate for production systems.
- Microsoft SQL Server is not a supported option.
- Oracle DB is the de facto standard at large enterprises and is fully supported with ownCloud Enterprise Edition only.

4.2.7 File Storage

While many customers are starting with NFS, sooner or later that requires scale-out storage. Currently the options are GPFS or GlusterFS, or an object store protocol like S3 (supported in Enterprise Edition only) or Swift. S3 also allows access to Ceph Storage.

4.2.8 Session Storage

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

4.2.9 References

Database High Availability

Performance enhancements for Apache and PHP

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

4.3 Preferred Linux Installation Method

For production environments, we recommend the installation from the tar archive. This applies in particular to scenarios, where the Web server, storage and database are on separate machines. In this constellation, all dependencies and requirements are managed by the package management of your operating system, while the ownCloud code itself is maintained in a sequence of simple steps as documented in our instructions for the *Manual Installation on Linux* and the *Manual ownCloud Upgrade*

The package installation is for single-server setups only.

4.3.1 Changes in ownCloud 9

Linux distribution packages (from Open Build Service) have been divided into multiple packages for ownCloud 9: owncloud_deps and owncloud_files.

Install the metapackage owncloud to get a complete installation with all dependencies.

The owncloud-files package installs only ownCloud, without Apache, database, or PHP dependencies.

The owncloud-deps packages install all dependencies: Apache, PHP, and MySQL. owncloud-deps is not intended to be installed by itself, but rather is pulled in by the metapackage owncloud.

owncloud-files is available for the following distributions, but not owncloud-deps.

You will have to install your own LAMP stack first. This allows you to create your own custom LAMP stack without dependency conflicts with the ownCloud package. Browse http://download.owncloud.org/download/repositories/9.1/owncloud/ to find the owncloud-files package for your distro:

- Ubuntu 14.04, 16.04
- Debian 7, 8
- RHEL 6, 7
- CentOS 6 SCL, 7
- SLES 12, 12 SP1
- openSUSE 13.2, Leap 42.1

ownCloud packages with dependencies are available for the following Linux distributions:

- Ubuntu 14.04, 16.04
- Debian 8
- RHEL 7
- CentOS 7
- SLES 12
- openSUSE 13.2, Leap 42.1

Repositories for Fedora, openSUSE Tumbleweed and Ubuntu 15.04 were dropped. If you use Fedora, use the tar archive with your own LAMP stack. openSUSE users can rely on LEAP packages for Tumbleweed.

Follow the instructions on the download page to install ownCloud. Then run the Installation Wizard to complete your installation. (see *Installation Wizard*).

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

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

4.3.2 Repos: Stable or Major Release?

You may use either of the following repositories for ownCloud 9.0:

- https://download.owncloud.org/download/repositories/stable/owncloud/
- https://download.owncloud.org/download/repositories/9.0/owncloud/

When you use the Stable repo, you never have to change it as it always tracks the current stable ownCloud version through all major releases: 8.2, 9.0, and so on. (Major releases are indicated by the second number, so 8.0, 8.1, 8.2, and 9.0 were all major releases.)

If you wish to track a specific major release, such as 9.0 or 9.1, then use that repo. That way you won't accidentally find yourself looking at an upgrade to the next major release before you're ready.

4.3.3 Installing ownCloud Enterprise Edition

See Installing & Upgrading ownCloud Enterprise Edition for instructions on installing ownCloud Enterprise edition.

4.3.4 Downgrading Not Supported

Downgrading is not supported and risks corrupting your data! If you want to revert to an older ownCloud version, install it from scratch and then restore your data from backup. Before doing this, file a support ticket (if you have paid support) or ask for help in the ownCloud forums to see if your issue can be resolved without downgrading.

4.3.5 BINLOG_FORMAT = STATEMENT

If your ownCloud installation fails and you see this in your ownCloud log:

An unhandled exception has been thrown: exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging. InnoDB is limited to row-logging when transaction isolation level is READ COMMITTED or READ UNCOMMITTED.'

See *MySQL / MariaDB with Binary Logging Enabled*.

4.3.6 Additional Installation Guides and Notes

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

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

If your distribution is not listed, your Linux distribution may maintain its own ownCloud packages, or you may prefer to install from source code (see *Manual Installation on Linux*).

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

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

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

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

4.4 Installation Wizard

4.4.1 Quick Start

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

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

Create an admin account				
molly				
⊘				
Storage & database 🕶				
Finish setup				
i Need help? See the documentation A				

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

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

- Data Directory Location
- Database Choice
- Trusted Domains
- Setting Strong Permissions

4.4.2 Data Directory Location

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

You should locate your ownCloud data directory outside of your Web root if you are using an HTTP server other than Apache, or you may wish to store your ownCloud data in a different location for other reasons (e.g. on a storage

Storage & database 👻						
Data folder						
	/var/o	oc_data				
_		Configure the datal	base			
	SQLite	MySQL/MariaDB	PostgreSQL			
	_					
	root					
	•••••		۲			
ocdb						
localhost						
Finish setup						

server). It is best to configure your data directory location at installation, as it is difficult to move after installation. You may put it anywhere; in this example is it located in /var/oc_data. This directory must already exist, and must be owned by your HTTP user (see *Setting Strong Directory Permissions*).

4.4.3 Database Choice

When installing ownCloud Server & ownCloud Enterprise editions the administrator may choose one of 3 supported database products.

SQLite

Is the default database for ownCloud Server, but is not available and not supported for the ownCloud Enterprise edition.

SQLite will be installed by the ownCloud packages and all the necessary dependencies will be satisfied. See see *Manual Installation on Linux* for a detailed listing of required and optional PHP modules.

If you used the packages to install ownCloud, you may "Finish Setup" with no additional steps to configure ownCloud using the SQLite database for limited use.

Please note that SQLite is good only for testing and lightweight single user setups. There is no client synchronization support. Therefore, other devices will not be able to synchronize with the data stored in an ownCloud SQLite database.

MYSQL/MariaDB

Is the ownCloud recommended database. See *MySQL/MariaDB*. It may be used with either ownCloud Server or ownCloud Enterprise editions.

First you should install the recommended MySQL/MariaDB database. Use package: sudo apt-get

```
install mariadb-server
```

If you have an administrator login that has permissions to create and modify databases, you may choose "Storage & Database". Then enter your database administrator name, password and any name you want for your ownCloud database.

Otherwise, use these steps to create temporary database administrator account.

```
sudo mysql --user=root mysql
CREATE USER 'dbadmin'@'localhost' IDENTIFIED BY 'Apassword';
GRANT ALL PRIVILEGES ON *.* TO 'dbadmin'@'localhost' WITH GRANT
OPTION;
FLUSH PRIVILEGES;
exit
```

PostgreSQL

Is also supported by ownCloud.

```
To install PostgreSQL, use the apt-get (or other apt-driving) command: sudo apt-get install postgresql
```

You may view more information about the PostgreSQL database system at: http://www.postgresql.org

In order to allow ownCloud access to the database, create a known password for the default user "postgres" added when the database is installed.

```
sudo -i -u postgres psql
postgres=# \password
Enter new password:
Enter it again:
postgres=# \q
exit
```

Oracle11g

Is only supported for the ownCloud Enterprise edition.

Database Setup By ownCloud

Your database and PHP connectors must be installed before you run the Installation Wizard by clicking the "Finish setup" button.

After you enter your temporary or root administrator login for your database, the installer creates a special database user with privileges limited to the ownCloud database. Then ownCloud needs only this special ownCloud database user and drops the temporary or root database login.

This new user is named from your ownCloud admin user, with an oc_prefix, and then given a random password. The ownCloud database user and password are written into config.php:

```
| For MySQL/MariaDB:
| ``'dbuser' => 'oc_dbadmin',``
| ``'dbpassword' => 'pX65Ty5DrHQkYPE5HRsDvyFHlZZHcm',``
|
| For PostgreSQL:
| ``'dbuser' => 'oc_postgres',``
| ``'dbpassword' => 'pX65Ty5DrHQkYPE5HRsDvyFHlZZHcm',``
```

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



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

4.4.4 Trusted Domains

All URLs used to access your ownCloud server must be whitelisted in your config.php file, under the trusted_domains setting. Users are allowed to log into ownCloud only when they point their browsers to a URL that is listed in the trusted_domains setting. You may use IP addresses and domain names. A typical configuration looks like this:

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

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



4.4.5 Setting Strong Directory Permissions

For hardened security we recommend setting the permissions on your ownCloud directories as strictly as possible, and for proper server operations. This should be done immediately after the initial installation and before running the setup. Your HTTP user must own the config/, data/ and apps/ directories so that you can configure ownCloud, create, modify and delete your data files, and install apps via the ownCloud Web interface.

You can find your HTTP user in your HTTP server configuration files. Or you can use *PHP Version and Information* (Look for the **User/Group** line).

- The HTTP user and group in Debian/Ubuntu is www-data.
- The HTTP user and group in Fedora/CentOS is apache.
- The HTTP user and group in Arch Linux is http.
- The HTTP user in openSUSE is wwwrun, and the HTTP group is www.

Note: When using an NFS mount for the data directory, do not change its ownership from the default. The simple act of mounting the drive will set proper permissions for ownCloud to write to the directory. Changing ownership as above could result in some issues if the NFS mount is lost.

The easy way to set the correct permissions is to copy and run this script. Replace the ocpath variable with the path to your ownCloud directory, and replace the htuser and htgroup variables with your HTTP user and group:

```
#!/bin/bash
ocpath='/var/www/owncloud'
htuser='www-data'
htgroup='www-data'
rootuser='root'
printf "Creating possible missing Directories\n"
mkdir -p $ocpath/data
mkdir -p $ocpath/assets
mkdir -p $ocpath/updater
```

```
printf "chmod Files and Directories\n"
find ${ocpath}/ -type f -print0 | xargs -0 chmod 0640
find ${ocpath}/ -type d -print0 | xargs -0 chmod 0750
printf "chown Directories\n"
chown -R ${rootuser}:${htgroup} ${ocpath}/
chown -R ${htuser}:${htgroup} ${ocpath}/apps/
chown -R ${htuser}:${htgroup} ${ocpath}/assets/
chown -R ${htuser}:${htgroup} ${ocpath}/config/
chown -R ${htuser}:${htgroup} ${ocpath}/data/
chown -R ${htuser}:${htgroup} ${ocpath}/themes/
chown -R ${htuser}:${htgroup} ${ocpath}/updater/
chmod +x ${ocpath}/occ
printf "chmod/chown .htaccess\n"
if [ -f ${ocpath}/.htaccess ]
then
 chmod 0644 ${ocpath}/.htaccess
 chown ${rootuser}:${htgroup} ${ocpath}/.htaccess
fi
if [ -f ${ocpath}/data/.htaccess ]
then
 chmod 0644 ${ocpath}/data/.htaccess
 chown ${rootuser}:${htgroup} ${ocpath}/data/.htaccess
fi
```

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

This lists the recommended modes and ownership for your ownCloud directories and files:

- All files should be read-write for the file owner, read-only for the group owner, and zero for the world
- All directories should be executable (because directories always need the executable bit set), read-write for the directory owner, and read-only for the group owner
- The apps/directory should be owned by [HTTP user]: [HTTP group]
- The config/ directory should be owned by [HTTP user]: [HTTP group]
- The themes / directory should be owned by [HTTP user]: [HTTP group]
- The assets/ directory should be owned by [HTTP user]: [HTTP group]
- The data/ directory should be owned by [HTTP user]: [HTTP group]
- The [ocpath]/.htaccess file should be owned by root: [HTTP group]
- The data/.htaccess file should be owned by root: [HTTP group]
- · Both . htaccess files are read-write file owner, read-only group and world

These strong permissions prevent upgrading your ownCloud server; see *Setting Permissions for Updating* for a script to quickly change permissions to allow upgrading.

4.5 Installing ownCloud From the Command Line

It is now possible to install ownCloud entirely from the command line. This is convenient for scripted operations, headless servers, and sysadmins who prefer the command line. There are three stages to installing ownCloud via the
command line:

1. Download and install the ownCloud code via your package manager, or download and unpack the tarball in the appropriate directories. (See *Preferred Linux Installation Method* and *Manual Installation on Linux*.)

2. Change the ownership of your owneloud directory to your HTTP user, like this example for Debian/Ubuntu. You must run occ as your HTTP user; see *Run occ As Your HTTP User*:

\$ sudo chown -R www-data:www-data /var/www/owncloud/

3. Use the occ command to complete your installation. This takes the place of running the graphical Installation Wizard:

```
$ cd /var/www/owncloud/
$ sudo -u www-data php occ maintenance:install --database
"mysql" --database-name "owncloud" --database-user "root" --database-pass
"password" --admin-user "admin" --admin-pass "password"
ownCloud is not installed - only a limited number of commands are available
ownCloud was successfully installed
```

Note that you must change to the root ownCloud directory, as in the example above, to run occ maintenance:install, or the installation will fail with a PHP fatal error message.

Supported databases are:

```
sqlite (SQLite3 - ownCloud Community edition only)
mysql (MySQL/MariaDB)
pgsql (PostgreSQL)
oci (Oracle - ownCloud Enterprise edition only)
```

See Command Line Installation for more information.

Finally, apply the correct strong permissions to your ownCloud files and directories (see *Setting Strong Directory Permissions*). This is an extremely important step. It helps protect your ownCloud installation, and ensures that it will run correctly.

4.5.1 BINLOG_FORMAT = STATEMENT

If your ownCloud installation fails and you see this in your ownCloud log:

```
An unhandled exception has been thrown: exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging. InnoDB is limited to row-logging when transaction isolation level is READ COMMITTED or READ UNCOMMITTED.'
```

See *MySQL / MariaDB with Binary Logging Enabled*.

4.6 Changing the Web Route

This admin manual assumes that the owncloud server shall be accessible under the web route /owncloud – this is also where the Linux packages make the server appear. You can change this in your Web server configuration, for example from https://example.com/owncloud/ to https://example.com/.

Basic system administrator and Apache configuration knowledge is prerequisite. Several configuration files need to be kept in sync when changing the Web route location.

On an Ubuntu-14.04 system the following files are typically involved:

- /etc/apache2/conf-enabled/owncloud.conf
- /var/www/owncloud/config/config.php
- /var/www/owncloud/.htaccess

4.6.1 Example: Moving from /owncloud to /

Edit the file /etc/apache2/conf-enabled/owncloud.conf to say:

```
Alias / "/var/www/owncloud/"
```

Edit /var/www/owncloud/config/config.php to say:

```
'overwrite.cli.url' => 'http://localhost/',
```

Edit the file /var/www/owncloud/.htaccess to say:

```
#### DO NOT CHANGE ANYTHING ABOVE THIS LINE ####
...
<IfModule mod_rewrite.c>
    RewriteBase /
...
```

Optionally also set your document root, though this is generally not needed or recommended. Edit the file /etc/apache2/sites-enabled/000-default.conf to say:

DocumentRoot /var/www/owncloud

Note: Since owncloud version 9.0.2 we support short URLs without index.php. The rewrite mechanisms involved a RewriteBase rule in .htaccess which is auto-generated when owncloud is first started. Depending on the exact way owncloud was installed (upgrade or fresh, plain tar archive, or packages) you may or may not find a RewriteBase in your .htaccess files. If it is not yet there, make sure to double check once the ownCloud server is up and running.

4.7 Installing and Managing Apps

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

4.7.1 Supported Apps

See Supported Apps in ownCloud for a list of supported Enterprise edition apps.

4.7.2 Viewing Enabled Apps

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

You will see which apps are enabled, not enabled, and recommended. You'll also see additional filters, such as Multimedia, Productivity, and Tool for finding more apps quickly.



4.7.3 Managing Apps

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

4.7.4 Adding Third Party Apps

Some apps are developed and supported by ownCloud directly. These have an **Official** tag. Apps with the **Approved** tag are community-developed and supported; they are maintained by trusted developers, and are under active development. Only **Official** and **Approved** apps are linked on the Apps page by default.

Click the app name to view a description of the app and any of the app settings in the Application View field. Clicking the **Enable** button will enable the app. If the app is not part of the ownCloud installation, it will be downloaded from the app store, installed and enabled.

Click the gear icon on the lower left to browse experimental apps in the ownCloud Apps Store. Install experimental apps at your own risk.

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

Note: If you would like to create or add your own ownCloud app, please refer to the developer manual.

4.7.5 Using Custom App Directories

Use the **apps_paths** array in config.php to set any custom apps directory locations. The key **path** defines the absolute file system path to the app folder. The key **url** defines the HTTP web path to that folder, starting at the

ownCloud web root. The key writable indicates if a user can install apps in that folder.

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

4.7.6 Using Your Own Appstore

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

To enable installation from your own apps store:

1. Set the appstoreenabled parameter to "true".

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

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

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

<?php

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

4.8 Supported Apps in ownCloud

4.8.1 AGPL Apps

- Activity
- AntiVirus
- Collaborative Tags
- Comments
- Encryption
- · External Sites
- External Storage

- ownCloud WebDAV Endpoint (handles old and new webdav endpoints)
- Federated File Sharing (allows file sharing across ownCloud instances)
- Federation (allows usernname auto-complete across ownCloud instances)
- Files (cannot be disabled)
- Files PDF Viewer
- · Files Sharing
- · Files TextEditor
- Files Trashbin
- · Files Versions
- Files VideoPlayer
- First Run Wizard
- Gallery
- Notifications
- Object Storage (Swift)
- Provisioning API
- Template Editor (for notification emails)
- Update Notifications
- User External
- User LDAP

4.8.2 Enterprise-Only Apps

- Enterprise License Key
- Files Drop
- File Firewall
- LDAP Home Connector
- Log user and Sharing actions (1 new app, replacing the 2 former logging apps)
- Object Storage (S3)
- SharePoint
- Shibboleth (SAML)
- Windows Network Drives (requires External Storage)
- Workflow

4.9 Manual Installation on Linux

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

Note: Enterprise customers should refer to Installing & Upgrading ownCloud Enterprise Edition

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

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

Note: Admins of SELinux-enabled distributions such as CentOS, Fedora, and Red Hat Enterprise Linux may need to set new rules to enable installing ownCloud. See *SELinux Configuration Tips* for a suggested configuration.

4.9.1 Prerequisites

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

Required:

- php5 (>= 5.4)
- PHP module ctype
- PHP module dom
- PHP module GD
- PHP module iconv
- PHP module JSON
- PHP module libxml (Linux package libxml2 must be >=2.7.0)
- PHP module mb multibyte
- PHP module posix
- PHP module SimpleXML
- PHP module XMLWriter

- PHP module zip
- PHP module zlib

Database connectors (pick the one for your database:)

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

Recommended packages:

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

Required for specific apps:

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

Note: SMB/Windows Network Drive mounts require the PHP module smbclient; see SMB/CIFS.

Recommended for specific apps (optional):

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

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

- PHP module apc
- PHP module apcu
- PHP module memcached
- PHP module redis (>= 2.2.6+, required for Transactional File Locking)

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

For preview generation (optional):

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

For command line processing (optional):

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

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

4.9.2 Example Installation on Ubuntu 14.04 LTS Server

Note: See Manual installation details for multiple distros, ownCloud 9.0 and 9.1 for installation hints for RHEL 7.2 and SLES 12.

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

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

- This installs the packages for the ownCloud core system. libapache2-mod-php5 provides the following PHP extensions: bcmath bz2 calendar Core ctype date dba dom ereg exif fileinfo filter ftp gettext hash iconv libxml mbstring mhash openssl pcre Phar posix Reflection session shmop SimpleXML soap sockets SPL standard sysvmsg sysvsem sysvshm tokenizer wddx xml xmlreader xmlwriter zip zlib. If you are planning on running additional apps, keep in mind that they might require additional packages. See *Prerequisites* for details.
- At the installation of the MySQL/MariaDB server, you will be prompted to create a root password. Be sure to remember your password as you will need it during ownCloud database setup.

Now download the archive of the latest ownCloud version:

- Go to the ownCloud Download Page.
- Go to Download ownCloud Server > Download > Archive file for server owners and download either the tar.bz2 or .zip archive.
- This downloads a file named owncloud-x.y.z.tar.bz2 or owncloud-x.y.z.zip (where x.y.z is the version number).
- Download its corresponding checksum file, e.g. owncloud-x.y.z.tar.bz2.md5, or owncloud-x.y.z.tar.bz2.sha256.
- Verify the MD5 or SHA256 sum:

```
md5sum -c owncloud-x.y.z.tar.bz2.md5 < owncloud-x.y.z.tar.bz2
sha256sum -c owncloud-x.y.z.tar.bz2.sha256 < owncloud-x.y.z.tar.bz2
md5sum -c owncloud-x.y.z.zip.md5 < owncloud-x.y.z.zip
sha256sum -c owncloud-x.y.z.zip.sha256 < owncloud-x.y.z.zip</pre>
```

• You may also verify the PGP signature:

```
wget https://download.owncloud.org/community/owncloud-x.y.z.tar.bz2.asc
wget https://owncloud.org/owncloud.asc
gpg --import owncloud.asc
gpg --verify owncloud-x.y.z.tar.bz2.asc owncloud-x.y.z.tar.bz2
```

• Now you can extract the archive contents. Run the appropriate unpacking command for your archive type:

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

• This unpacks to a single owncloud directory. Copy the ownCloud directory to its final destination. When you are running the Apache HTTP server you may safely install ownCloud in your Apache document root:

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

where /path/to/webserver/document-root is replaced by the document root of your Web server:

cp -r owncloud /var/www

On other HTTP servers it is recommended to install ownCloud outside of the document root.

4.9.3 BINLOG_FORMAT = STATEMENT

If your ownCloud installation fails and you see this in your ownCloud log:

An unhandled exception has been thrown: exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging. InnoDB is limited to row-logging when transaction isolation level is READ COMMITTED or READ UNCOMMITTED.'

See *MySQL / MariaDB with Binary Logging Enabled*.

4.9.4 Apache Web Server Configuration

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

```
Alias /owncloud "/var/www/owncloud/"
<Directory /var/www/owncloud/>
Options +FollowSymlinks
AllowOverride All
<IfModule mod_dav.c>
Dav off
</IfModule>
SetEnv HOME /var/www/owncloud
SetEnv HTTP_HOME /var/www/owncloud
```

</Directory>

Then create a symlink to /etc/apache2/sites-enabled:

```
ln -s /etc/apache2/sites-available/owncloud.conf /etc/apache2/sites-enabled/owncloud.conf
```

Additional Apache Configurations

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

a2enmod rewrite

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

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

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

a2enmod setenvif

• You must disable any server-configured authentication for ownCloud, as it uses Basic authentication internally for DAV services. If you have turned on authentication on a parent folder (via e.g. an AuthType Basic directive), you can turn off the authentication specifically for the ownCloud entry. Following the above example configuration file, add the following line in the <Directory section:

Satisfy Any

- When using SSL, take special note of the ServerName. You should specify one in the server configuration, as well as in the CommonName field of the certificate. If you want your ownCloud to be reachable via the internet, then set both of these to the domain you want to reach your ownCloud server.
- Now restart Apache:

service apache2 restart

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

4.9.5 Enabling SSL

Note: You can use ownCloud over plain HTTP, but we strongly encourage you to use SSL/TLS to encrypt all of your server traffic, and to protect user's logins and data in transit.

Apache installed under Ubuntu comes already set-up with a simple self-signed certificate. All you have to do is to enable the ssl module and the default site. Open a terminal and run:

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

Note: See Security (Import SSL Certificates) for help on managing self-signed certificates.

4.9.6 Installation Wizard

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

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

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

To use occ see Installing ownCloud From the Command Line.

To use the graphical Installation Wizard see Installation Wizard.

4.9.7 Setting Strong Directory Permissions

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

Now your ownCloud server is ready to use.

4.9.8 SELinux Configuration Tips

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

4.9.9 php.ini Configuration Notes

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

php.ini - used by the Web server:

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

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

/etc/php5/cli/php.ini

4.9.10 php-fpm Configuration Notes

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

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

System environment variables

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

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

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

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

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

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

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

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

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

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

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

Maximum upload size

If you want to increase the maximum upload size, you will also have to modify your php-fpm configuration and increase the upload_max_filesize and post_max_size values. You will need to restart php5-fpm and your HTTP server in order for these changes to be applied.

.htaccess notes for Apache

ownCloud comes with its own owncloud/.htaccess file. Because php-fpm can't read PHP settings in .htaccess these settings and permissions must be set in the owncloud/.user.ini file.

4.9.11 Other Web Servers

nginx Example Configurations Other HTTP servers Univention Corporate Server installation

4.10 ownCloud Community Appliance

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

4.10.1 Instructions for VirtualBox and OVA

Follow these steps to get the appliance working:

- 1. Download the Virtual Machine image zip file and unpack it.
- 2. Start VirtualBox and click on File ... > Import Appliance and import your new ownCloud image.
- 3. Click the green Start arrow. After a minute you should see the console greeting message.
- 4. Note the username and password here. It is a random password that we generate for you on first boot. If you log in at the console, you'll be prompted to change the password. This is optional.
- 5. With your Web browser try http://localhost:8888 or http://localhost:80 or the address printed on the console. One of them should work. If not, please review and adjust the network setup of Virtualbox to bridged mode.

| 1 | 🦻 xUbuntu_14.04-owncloud-8.1.1-1-201509081909 [wird ausgeführt] - Oracle VM VirtualBox _ 🔤 🗆 🗙 | | |
|-----------------|---|--------|--|
| Date | i Maschine Anzeige Eingabe Geräte Hilfe | | |
| | Ubuntu 14.04.2 LTS owncloud tty1 | + | |
| | Welcome to ownCloud! # Version: 8.1.1 | | |
| | This server is reachable at https://10.0.2.15/owncloud
Initial admin login: admin
Initial admin password: m1e4q8x0k6 | | |
| | If the virtual machine is run with NAT, please review the port
forwarding of the network adapter, or try http://localhost:8888 | | |
| | You can now logon to your ownCloud by using the ip–address from
above with your web browser. Please import the SSL cert to your
browser, or accept the security warning to connect to your
ownCloud via HTTPS. | | |
| | OPTIONAL:
If you want to do the final setup (e.g. change admin password),
please log in as user 'admin' to run the setup–script. | + | |
| owncloud login: | | | |
| | 🔊 🗗 🚍 🖳 🛞 🏈 Strg | Rechts | |

6. You should see a Web page with login credentials (if you haven't changed them already) and a list of URLs to try to reach the ownCloud web service. Which one works, again depends on the network setup of your hypervisor.

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

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

For VMware

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

4.10.2 Software Appliances

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

- Tech and Me ownCloud VM on Ubuntu 16.04 with PHP 7, MySQL, and Apache, fully configured environment.
- SUSE Studio, ownCloud on openSuSE, which runs directly from an USB stick.
- · Amahi home server



Figure 4.1: Click to enlarge

4.11 Installing PHP 5.4 on RHEL 6 and CentOS 6

Red Hat Enterprise Linux and CentOS 6 still ship with PHP 5.3. ownCloud requires PHP 5.4 or better. There are several third-party repositories that supply PHP 5.4, but you must use the Software Collections (SCL) repository to be in compliance with your RHEL support contract, and not any other third-party repository.

4.11.1 RHEL 6

Follow these steps to install PHP 5.4 from SCL. First you must use your Subscription Manager to enable SCL:

subscription-manager repos --enable rhel-server-rhscl-6-eus-rpms

Then install PHP 5.4 and these modules:

yum install php54 php54-php php54-php-gd php54-php-mbstring

You must also install the updated database module for your database. This example installs the new PHP 5.4 module for MySQL/MariaDB:

yum install php54-php-mysqlnd

Disable loading the old PHP 5.3 Apache module:

mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php53.off

You should now have a /etc/httpd/conf.d/php54-php.conf file, which loads the correct PHP 5.4 module for Apache.

Then restart Apache:

service httpd restart

Verify with PHP Version and Information that your Apache server is using PHP 5.4 and loading the correct modules.

4.11.2 CentOS 6

First install the SCL repo:

yum install centos-release-SCL

Then install PHP 5.4 and these modules:

yum install php54 php54-php php54-php-gd php54-php-mbstring

You must also install the updated database module. This installs the new PHP 5.4 module for MySQL/MariaDB:

yum install php54-php-mysqlnd

Disable loading the old PHP 5.3 Apache module:

mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php53.off

You should now have a /etc/httpd/conf.d/php54-php.conf file, which loads the correct PHP 5.4 module for Apache.

Finally, restart Apache:

service httpd restart

Verify with PHP Version and Information that your Apache server is using PHP 5.4 and loading the correct modules.

4.12 Installing PHP 5.5 on RHEL 7 and CentOS 7

PHP 5.4 has been end-of-life since September 2015 and is no longer supported by the PHP team. RHEL 7 still ships with PHP 5.4, and Red Hat supports it. ownCloud also supports PHP 5.4, so upgrading is not required. However, it is highly recommended to upgrade to PHP 5.5+ for best security and performance.

Before upgrading, evaluate all of your PHP apps for compatibility with PHP 5.5.

4.12.1 RHEL 7 Upgrade to PHP 5.5

To upgrade to PHP 5.5, you must use the Software Collections (SCL) repository to be in compliance with your RHEL support contract, and not any other third-party repository. Follow these steps to install PHP 5.5 from SCL. First you must use your Subscription Manager to enable SCL:

subscription-manager repos --enable rhel-server-rhscl-7-eus-rpms

Then install PHP 5.5 and these modules:

yum install php55 php55-php php55-php-gd php55-php-mbstring

You must also install the updated database module for your database. This installs the new PHP 5.5 module for MySQL/MariaDB:

yum install php55-php-mysqlnd

If you are using the ownCloud LDAP app, you need this module:

yum install php55-php-ldap

Disable loading the old PHP Apache modules by changing their names:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php54.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php54.off
```

Copy the PHP 5.5 Apache modules into place:

cp /opt/rh/httpd24/root/etc/httpd/conf.d/php55-php.conf /etc/httpd/conf.d/ cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/10-php55-php.conf /etc/httpd/conf.modules.d/ cp /opt/rh/httpd24/root/etc/httpd/modules/libphp55-php5.so /etc/httpd/modules/

Then restart Apache:

service httpd restart

Verify with phpinfo that your Apache server is using PHP 5.5 and loading the correct modules; see *PHP Version* and *Information* to learn how to use phpinfo.

4.12.2 CentOS 7 Upgrade to PHP 5.5

To upgrade to PHP 5.5, use the Red Hat Software Collections (SCL) repository.

Before upgrading, evaluate all of your PHP apps for compatibility with PHP 5.5.

Follow these steps to install PHP 5.5 from SCL. First install the SCL repository:

yum install centos-release-scl

Then install PHP 5.5 and these modules:

yum install php55 php55-php php55-php-gd php55-php-mbstring

You must also install the updated database module for your database. This installs the new PHP 5.5 module for MySQL/MariaDB:

yum install php55-php-mysqlnd

If you are using the ownCloud LDAP app, you need this module:

yum install php55-php-ldap

Disable loading the old PHP Apache modules by changing their names:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php54.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php54.off
```

Copy the PHP 5.5 Apache modules into place:

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/php55-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/10-php55-php.conf /etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/libphp55-php5.so /etc/httpd/modules/
```

Then restart Apache:

service httpd restart

Verify with phpinfo that your Apache server is using PHP 5.5 and loading the correct modules; see *PHP Version* and *Information* to learn how to use phpinfo.

4.13 SELinux Configuration

When you have SELinux enabled on your Linux distribution, you may run into permissions problems after a new ownCloud installation, and see permission denied errors in your ownCloud logs.

When you are testing ownCloud or troubleshooting,

The following settings should work for most SELinux systems that use the default distro profiles. Run these commands as root, and remember to adjust the filepaths in these examples for your installation:

```
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/data(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/apps(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/assets(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/.htaccess'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/.htaccess'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/.htaccess'
```

```
restorecon -Rv '/var/www/html/owncloud/'
```

If you uninstall ownCloud you need to remove the ownCloud directory labels. To do this execute the following commands as root after uninstalling ownCloud:

```
semanage fcontext -d '/var/www/html/owncloud/data(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/config(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/apps(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/assets(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/.htaccess'
semanage fcontext -d '/var/www/html/owncloud/.user.ini'
```

```
restorecon -Rv '/var/www/html/owncloud/'
```

Note: The assets folder is only required if JavaScript and CSS Asset Management is enabled. (asset-pipeline.enabled' => true, in config.php)

If you have customized SELinux policies and these examples do not work, you must give the HTTP server write access to these directories:

```
/var/www/html/owncloud/data
/var/www/html/owncloud/config
/var/www/html/owncloud/apps
/var/www/html/owncloud/assets
```

4.13.1 Enable updates via the web interface

To enable updates via the ownCloud web interface, you may need this to enable writing to the ownCloud directories:

setsebool httpd_unified on

When the update is completed, disable write access:

```
setsebool -P httpd_unified off
```

4.13.2 Disallow write access to the whole web directory

For security reasons it's suggested to disable write access to all folders in /var/www/ (default):

setsebool -P httpd_unified off

4.13.3 Allow access to a remote database

An additional setting is needed if your installation is connecting to a remote database:

setsebool -P httpd_can_network_connect_db on

4.13.4 Allow access to LDAP server

Use this setting to allow LDAP connections: setsebool -P httpd_can_connect_ldap on

4.13.5 Allow access to remote network

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

setsebool -P httpd_can_network_connect on

4.13.6 Allow access to network memcache

This setting is not required if httpd_can_network_connect is already on:

setsebool -P httpd_can_network_memcache on

4.13.7 Allow access to SMTP/sendmail

If you want to allow ownCloud to send out e-mail notifications via sendmail you need to use the following setting:

setsebool -P httpd_can_sendmail on

4.13.8 Allow access to CIFS/SMB

If you have placed your datadir on a CIFS/SMB share use the following setting:

setsebool -P httpd_use_cifs on

4.13.9 Allow access to FuseFS

If your owncloud data folder resides on a Fuse Filesystem (e.g. EncFS etc), this setting is required as well:

setsebool -P httpd_use_fusefs on

4.13.10 Allow access to GPG for Rainloop

If you use a the rainloop webmail client app which supports GPG/PGP, you might need this:

setsebool -P httpd_use_gpg on

4.13.11 Troubleshooting

For general Troubleshooting of SELinux and its profiles try to install the package setroubleshoot and run:

```
sealert -a /var/log/audit/audit.log > /path/to/mylogfile.txt
```

to get a report which helps you configuring your SELinux profiles.

Another tool for troubleshooting is to enable a single ruleset for your ownCloud directory:

```
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud(/.*)?'
restorecon -RF /var/www/html/owncloud
```

It is much stronger security to have a more fine-grained ruleset as in the examples at the beginning, so use this only for testing and troubleshooting. It has a similar effect to disabling SELinux, so don't use it on production systems.

See this discussion on GitHub to learn more about configuring SELinux correctly for ownCloud.

4.14 nginx Example Configurations

This page covers example nginx configurations to use with running an ownCloud server. Note that nginx is not officially supported, and this page is community-maintained. (Thank you, contributors!)

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

4.14.1 Example Configurations

Be careful about line breaks if you copy the examples, as long lines may be broken for page formatting.

Thanks to @josh4trunks for providing / creating these configuration examples.

You can use ownCloud over plain http, but we strongly encourage you to use SSL/TLS to encrypt all of your server traffic, and to protect user's logins and data in transit.

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

ownCloud in the webroot of nginx

The following config should be used when ownCloud is placed in the webroot of your nginx installation.

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php5-fpm.sock;
}
server {
   listen 80;
    server_name cloud.example.com;
    # enforce https
    return 301 https://$server_name$request_uri;
}
server {
    listen 443 ssl;
    server_name cloud.example.com;
    ssl_certificate /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key /etc/ssl/nginx/cloud.example.com.key;
    # Add headers to serve security related headers
    # Before enabling Strict-Transport-Security headers please read into this topic first.
    #add_header Strict-Transport-Security "max-age=15552000; includeSubDomains";
    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    add_header X-Download-Options noopen;
    add_header X-Permitted-Cross-Domain-Policies none;
    # Path to the root of your installation
    root /var/www/owncloud/;
    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }
    # The following 2 rules are only needed for the user_webfinger app.
    # Uncomment it if you're planning to use this app.
    #rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
    #rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json last;
    location = /.well-known/carddav {
        return 301 $scheme://$host/remote.php/dav;
    }
    location = /.well-known/caldav {
        return 301 $scheme://$host/remote.php/dav;
    }
    location /.well-known/acme-challenge { }
    # set max upload size
    client_max_body_size 512M;
    fastcgi_buffers 64 4K;
```

```
# Disable gzip to avoid the removal of the ETag header
gzip off;
# Uncomment if your server is build with the ngx_pagespeed module
# This module is currently not supported.
#pagespeed off;
error_page 403 /core/templates/403.php;
error_page 404 /core/templates/404.php;
location / {
   rewrite ^ /index.php$uri;
}
location ~ ^/(?:build|tests|config|lib|3rdparty|templates|data)/ {
   return 404;
location ~ ^/(?:\.|autotest|occ|issue|indie|db_|console) {
   return 404;
}
location ~ ^/(?:index|remote|public|cron|core/ajax/update|status|ocs/v[12]|updater/.+|ocs-provide
   fastcgi_split_path_info ^(.+\.php)(/.*)$;
   include fastcgi_params;
   fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
   fastcgi_param PATH_INFO $fastcgi_path_info;
   fastcgi_param HTTPS on;
   fastcgi_param modHeadersAvailable true; #Avoid sending the security headers twice
   fastcgi_param front_controller_active true;
   fastcgi_pass php-handler;
   fastcgi_intercept_errors on;
   fastcqi_request_buffering off; #Available since nginx 1.7.11
}
location ~ ^/(?:updater|ocs-provider)(?:$|/) {
   try_files $uri $uri/ =404;
   index index.php;
}
# Adding the cache control header for js and css files
# Make sure it is BELOW the PHP block
location ~* \.(?:css|js)$ {
   try_files $uri /index.php$uri$is_args$args;
   add_header Cache-Control "public, max-age=7200";
    # Add headers to serve security related headers (It is intended to have those duplicated to t
    # Before enabling Strict-Transport-Security headers please read into this topic first.
   #add_header Strict-Transport-Security "max-age=15552000; includeSubDomains";
   add_header X-Content-Type-Options nosniff;
   add_header X-Frame-Options "SAMEORIGIN";
   add_header X-XSS-Protection "1; mode=block";
   add_header X-Robots-Tag none;
   add_header X-Download-Options noopen;
   add header X-Permitted-Cross-Domain-Policies none;
   # Optional: Don't log access to assets
   access_log off;
}
location ~* \.(?:svg|gif|png|html|ttf|woff|ico|jpg|jpeg)$ {
```

```
try_files $uri /index.php$uri$is_args$args;
    # Optional: Don't log access to other assets
    access_log off;
  }
}
```

ownCloud in a subdir of nginx

The following config should be used when ownCloud is placed within a subdir of your nginx installation.

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php5-fpm.sock;
}
server {
   listen 80;
   server_name cloud.example.com;
    # enforce https
   return 301 https://$server_name$request_uri;
}
server {
   listen 443 ssl;
    server_name cloud.example.com;
    ssl_certificate /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key /etc/ssl/nginx/cloud.example.com.key;
    # Add headers to serve security related headers
    # Before enabling Strict-Transport-Security headers please read into this topic first.
    #add_header Strict-Transport-Security "max-age=15552000; includeSubDomains";
    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    add_header X-Download-Options noopen;
    add_header X-Permitted-Cross-Domain-Policies none;
    # Path to the root of your installation
    root /var/www/;
    location = /robots.txt {
       allow all;
       log_not_found off;
        access_log off;
    }
    # The following 2 rules are only needed for the user_webfinger app.
    # Uncomment it if you're planning to use this app.
    #rewrite ^/.well-known/host-meta /owncloud/public.php?service=host-meta last;
    #rewrite ^/.well-known/host-meta.json /owncloud/public.php?service=host-meta-json last;
    location = /.well-known/carddav {
       return 301 $scheme://$host/owncloud/remote.php/dav;
    }
    location = /.well-known/caldav {
```

```
return 301 $scheme://$host/owncloud/remote.php/dav;
}
location /.well-known/acme-challenge { }
location ^~ /owncloud {
    # set max upload size
   client_max_body_size 512M;
    fastcqi_buffers 64 4K;
    # Disable gzip to avoid the removal of the ETag header
   gzip off;
    # Uncomment if your server is build with the ngx_pagespeed module
    # This module is currently not supported.
   #pagespeed off;
   error_page 403 /owncloud/core/templates/403.php;
   error_page 404 /owncloud/core/templates/404.php;
   location /owncloud {
       rewrite ^ /owncloud/index.php$uri;
    }
   location ~ ^/owncloud/(?:build|tests|config|lib|3rdparty|templates|data)/ {
        return 404;
   location ~ ^/owncloud/(?:\.lautotestlocclissuelindieldb_lconsole) {
       return 404;
    }
   location ~ ^/owncloud/(?:index|remote|public|cron|core/ajax/update|status|ocs/v[12]|updater/
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
       include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        fastcgi_param HTTPS on;
        fastcgi_param modHeadersAvailable true; #Avoid sending the security headers twice
        fastcgi_param front_controller_active true;
        fastcgi_pass php-handler;
        fastcgi_intercept_errors on;
        fastcgi_request_buffering off; #Available since nginx 1.7.11
    }
   location ~ ^/owncloud/(?:updater|ocs-provider)(?:$|/) {
        try_files $uri $uri/ =404;
       index index.php;
    }
    # Adding the cache control header for js and css files
    # Make sure it is BELOW the PHP block
   location ~* \.(?:css|js)$ {
        try_files $uri /owncloud/index.php$uri$is_args$args;
        add_header Cache-Control "public, max-age=7200";
        # Add headers to serve security related headers (It is intended to have those duplicated
        # Before enabling Strict-Transport-Security headers please read into this topic first.
        #add_header Strict-Transport-Security "max-age=15552000; includeSubDomains";
```

```
add_header X-Content-Type-Options nosniff;
add_header X-Frame-Options "SAMEORIGIN";
add_header X-XSS-Protection "1; mode=block";
add_header X-Robots-Tag none;
add_header X-Download-Options noopen;
add_header X-Permitted-Cross-Domain-Policies none;
# Optional: Don't log access to assets
access_log off;
}
location ~* \.(?:svg|gif|png|html|ttf|woff|ico|jpg|jpeg)$ {
try_files $uri /owncloud/index.php$uri$is_args$args;
# Optional: Don't log access to other assets
access_log off;
}
}
```

Suppressing Log Messages

}

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

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

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

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

This could be caused by the:

```
location \sim * \ (?:css|js) {
```

block shown above not located below the:

location ~ $\ (?:\$|/)$ {

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

Performance Tuning

nginx (<1.9.5) <ngx_http_spdy_module nginx (+1.9.5) <ngx_http_http2_module

To use http_v2 for nginx you have to check two things:

1.) be aware that this module is not built in by default due to a dependency to the openssl version used on your system. It will be enabled with the --with-http_v2_module configuration parameter during compilation. The dependency should be checked automatically. You can check the presence of http_v2 with nginx -V 2>&1 | grep http_v2 -o. An example of how to compile nginx can be found in section "Configure nginx with the nginx-cache-purge module" below.

2.) When you have used SPDY before, the nginx config has to be changed from listen 443 ssl spdy; to listen 443 ssl http2;

nginx: caching ownCloud gallery thumbnails

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

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

Note: Unlike Apache, nginx does not dynamically load modules. All modules needed must be compiled into nginx. This is one of the reasons for nginx's performance. It is expected to have an already running nginx installation with a working configuration set up as described in the ownCloud documentation.

nginx module check

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

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

If your output contains ngx_cache_purge, you can continue with the configuration, otherwise you need to manually compile nginx with the module needed.

Compile nginx with the nginx-cache-purge module

1. Preparation:

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

Add the following lines (if different, replace {trusty} by your distribution name):

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

Then run sudo apt-get update

Note: If you're not overly cautious and wish to install the latest and greatest nginx packages and features, you may have to install nginx from its mainline repository. From the nginx homepage: "In general, you should deploy nginx from its mainline branch at all times." If you would like to use standard nginx from the latest mainline branch but without compiling in any additional modules, just run sudo apt-get install nginx.

2. Download the nginx source from the ppa repository

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

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

ls -la

Please replace {release} with the release downloaded:

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

If folder "modules" is not present, do:

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

If not present, add the following line at the top under:

```
#export DH_VERBOSE=1:
MODULESDIR = $(CURDIR)/debian/modules
```

And at the end of every configure command add:

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

Don't forget to escape preceeding lines with a backslash $\$. The parameters may now look like:

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

4. Compile and install nginx

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

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

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

It should now show: ngx_cache_purge

Show nginx version including all features compiled and installed:

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

6. Mark nginx to be blocked from further updates via apt-get

sudo dpkg --get-selections | grep nginx

For every nginx component listed run sudo apt-mark hold <component>

7. Regular checks for nginx updates

Do a regular visit on the nginx news page and proceed in case of updates with items 2 to 5.

Configure nginx with the nginx-cache-purge module

1. **Preparation** Create a directory where nginx will save the cached thumbnails. Use any path that fits to your environment. Replace {path} in this example with your path created:

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

2. Configuration

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

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

```
# cache_purge
fastcgi_cache_path {path} levels=1:2 keys_zone=OWNCLOUD:100m inactive=60m;
map $request_uri $skip_cache {
    default 1;
    ~*/thumbnail.php 0;
    ~*/apps/galleryplus/ 0;
    ~*/apps/galleryplus/ 0;
}
```

Note: Please adopt or delete any regex line in the map block according your needs and the ownCloud version used. As an alternative to mapping, you can use as many if statements in your server block as necessary:

```
set $skip_cache 1;
if ($request_uri ~* "thumbnail.php") { set $skip_cache 0; }
if ($request_uri ~* "/apps/galleryplus/") { set $skip_cache 0; }
if ($request_uri ~* "/apps/gallery/") { set $skip_cache 0; }
```

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

```
# cache_purge (with $http_cookies we have unique keys for the user)
fastcgi_cache_key $http_cookie$request_method$host$request_uri;
fastcqi_cache_use_stale error timeout invalid_header http_500;
fastcgi_ignore_headers Cache-Control Expires Set-Cookie;
location ~ \phi(?:$/) {
      fastcgi_split_path_info ^(.+\.php)(/.+)$;
      include fastcgi_params;
      fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
      fastcgi_param PATH_INFO $fastcgi_path_info;
      fastcgi_param HTTPS on;
      fastcgi_pass php-handler;
      # cache_purge
      fastcgi_cache_bypass $skip_cache;
      fastcgi_no_cache $skip_cache;
      fastcgi_cache OWNCLOUD;
      fastcqi_cache_valid 60m;
      fastcgi_cache_methods GET HEAD;
```

Note: Note regarding the fastcgi_pass parameter: Use whatever fits your configuration. In the example above, an upstream was defined in an nginx global configuration file. This may look like:

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

3. Test the configuration

sudo nginx -s reload

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

CHAPTER

FIVE

OWNCLOUD SERVER CONFIGURATION

5.1 Warnings on Admin Page

Your ownCloud server has a built-in configuration checker, and it reports its findings at the top of your Admin page. These are some of the warnings you might see, and what to do about them.

Security & setup warnings

- No memory cache has been configured. To enhance your performance please configure a memcache if available. Further information can be found in our documentation.
- You are accessing this site via HTTP. We strongly suggest you configure your server to require using HTTPS instead.

Please double check the installation guides ≯, and check for any errors or warnings in the log.

5.1.1 Cache Warnings

"No memory cache has been configured. To enhance your performance please configure a memcache if available." ownCloud supports multiple php caching extentions:

- APC (PHP 5.4 only)
- APCu (PHP 5.5+, minimum required PHP extension version 4.0.6)
- Memcached
- Redis (minimum required PHP extension version: 2.2.6)

You will see this warning if you have no caches installed and enabled, or if your cache does not have the required minimum version installed; older versions are disabled because of performance problems.

If you see "*{Cache}* below version *{Version}* is installed. for stability and performance reasons we recommend to update to a newer *{Cache}* version" then you need to upgrade, or, if you're not using it, remove it.

You are not required to use any caches, but caches improve server performance. See Configuring Memory Caching.

5.1.2 Transactional file locking is disabled

"Transactional file locking is disabled, this might lead to issues with race conditions."

Please see Transactional File Locking on how to correctly configure your environment for transactional file locking.

5.1.3 You are accessing this site via HTTP

"You are accessing this site via HTTP. We strongly suggest you configure your server to require using HTTPS instead." Please take this warning seriously; using HTTPS is a fundamental security measure. You must configure your Web server to support it, and then there are some settings in the **Security** section of your ownCloud Admin page to enable. The following pages describe how to enable HTTPS on the Apache and Nginx Web servers.

Enabling SSL (on Apache)

Use HTTPS

nginx Example Configurations

5.1.4 The test with getenv("PATH") only returns an empty response

Some environments are not passing a valid PATH variable to ownCloud. The *php-fpm Configuration Notes* provides the information about how to configure your environment.

5.1.5 The "Strict-Transport-Security" HTTP header is not configured

"The "Strict-Transport-Security" HTTP header is not configured to least "15768000" seconds. For enhanced security we recommend enabling HSTS as described in our security tips."

The HSTS header needs to be configured within your Web server by following the *Enable HTTP Strict Transport* Security documentation

5.1.6 /dev/urandom is not readable by PHP

"/dev/urandom is not readable by PHP which is highly discouraged for security reasons. Further information can be found in our documentation."

This message is another one which needs to be taken seriously. Please have a look at the *Give PHP read access to /dev/urandom* documentation.

5.1.7 Your Web server is not yet set up properly to allow file synchronization

"Your web server is not yet set up properly to allow file synchronization because the WebDAV interface seems to be broken."

At the ownCloud community forums a larger FAQ is maintained containing various information and debugging hints.

5.1.8 Outdated NSS / OpenSSL version

"cURL is using an outdated OpenSSL version (OpenSSL/\$version). Please update your operating system or features such as installing and updating apps via the app store or Federated Cloud Sharing will not work reliably."

"cURL is using an outdated NSS version (NSS/\$version). Please update your operating system or features such as installing and updating apps via the app store or Federated Cloud Sharing will not work reliably."

There are known bugs in older OpenSSL and NSS versions leading to misbehaviour in combination with remote hosts using SNI. A technology used by most of the HTTPS websites. To ensure that ownCloud will work properly you

need to update OpenSSL to at least 1.0.2b or 1.0.1d. For NSS the patch version depends on your distribution and an heuristic is running the test which actually reproduces the bug. There are distributions such as RHEL/CentOS which have this backport still pending.

5.1.9 Your Web server is not set up properly to resolve /.well-known/caldav/ or /.well-known/carddav/

Both URLs need to be correctly redirected to the DAV endpoint of ownCloud. Please refer to *Service discovery* for more info.

5.1.10 Some files have not passed the integrity check

Please refer to the Fixing Invalid Code Integrity Messages documentation how to debug this issue.

5.1.11 Your database does not run with "READ COMMITED" transaction isolation level

"Your database does not run with "READ COMMITED" transaction isolation level. This can cause problems when multiple actions are executed in parallel."

Please refer to *MySQL / MariaDB* "*READ COMMITED*" *transaction isolation level* how to configure your database for this requirement.

5.2 Importing System-wide and Personal SSL Certificates

Modern Web browsers try to keep us safe, and so they blast us with scary warnings when sites have the smallest errors in their SSL certificates, or when they use self-signed SSL certificates. ownCloud admins encounter this when creating Federation shares, or setting up external storage mounts. There is no reason against using self-signed certificates on your own networks; they're fast, free, and easy.

5.2.1 Importing Personal SSL Certificates

ownCloud has several methods for importing self-signed certificates so that you don't have to hassle with Web browser warnings. When you allow your users to create their own external storage mounts or Federation shares, they can import SSL certificates for those shares on their Personal pages.

SSL Root Certificates

Import root certificate

Click the **Import root certificate** button to open a file picker. You can distribute copies of your SSL certificates to your users (via an ownCloud share!), or users can download them from their Web browsers. Click on the little padlock icon and click through until you see a **View Certificate** button, then keep going until you can download it. In Firefox and Chromium there is an **Export** button for downloading your own copy of a site's SSL certificate.



Figure 5.1: Click "More information" in Firefox to import SSL certificate

5.2.2 Site-wide SSL Import

The personal imports only work for individual users. You can enable site-wide SSL certificates for all of your users on your ownCloud admin page. To enable this, you must add this line to your config.php file:

'enable_certificate_management' => true,

Then you'll have a **Import root certificate** button on your admin page, just like the one on your personal page.

5.2.3 Using OCC to Import and Manage SSL Certificates

The occ command has options for listing and managing your SSL certificates:

```
security:certificates list trusted certificates
security:certificates:remove remove trusted certificate
```

See Using the occ Command to learn about how to use occ.

5.3 Using the occ Command

ownCloud's occ command (ownCloud console) is ownCloud's command-line interface. You can perform many common server operations with occ, such as installing and upgrading ownCloud, manage users, encryption, passwords, LDAP setting, and more.

occ is in the owncloud/ directory; for example /var/www/owncloud on Ubuntu Linux. occ is a PHP script. You must run it as your HTTP user to ensure that the correct permissions are maintained on your ownCloud files and directories. In ownCloud 8.2+ you may run it from any directory (specifying the filepath); in previous releases it had to be run from the owncloud/ directory.

5.3.1 occ Command Directory

- Run occ As Your HTTP User
- Apps Commands
- Background Jobs Selector

- Config Commands
- Dav Commands
- Database Conversion
- Encryption
- Federation Sync
- File Operations
- Files External
- Integrity Check
- 110n, Create Javascript Translation Files for Apps
- LDAP Commands
- Logging Commands
- Maintenance Commands
- Security (Import SSL Certificates)
- Shibboleth Modes (Enterprise Edition only)
- Trashbin
- User Commands
- Versions
- Command Line Installation
- Command Line Upgrade

5.3.2 Run occ As Your HTTP User

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

- The HTTP user and group in Debian/Ubuntu is www-data.
- The HTTP user and group in Fedora/CentOS is apache.
- The HTTP user and group in Arch Linux is http.
- The HTTP user in openSUSE is wwwrun, and the HTTP group is www.

If your HTTP server is configured to use a different PHP version than the default (/usr/bin/php), occ should be run with the same version. For example, in CentOS 6.5 with SCL-PHP54 installed, the command looks like this:

sudo -u apache /opt/rh/php54/root/usr/bin/php /var/www/html/owncloud/occ

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

```
sudo -u www-data php occ
ownCloud version 9.0.0
Usage:
  command [options] [arguments]
Options:
  -h, --help Display this help message
```

| -q,quiet | Do not output any message |
|---------------------|--|
| -V,version | Display this application version |
| ansi | Force ANSI output |
| no-ansi | Disable ANSI output |
| -n,no-interaction | Do not ask any interactive question |
| no-warnings | Skip global warnings, show command output only |
| -v vv vvv,verbose | Increase the verbosity of messages: 1 for normal output, |
| | 2 for more verbose output and 3 for debug |
| | |
| Available commands: | |
| check | check dependencies of the server |
| | environment |
| help | Displays help for a command |
| list | Lists commands |
| status | show some status information |
| upgrade | run upgrade routines after installation of |
| | a new release. The release has to be |
| | installed before. |
| | |

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

Run it with the -h option for syntax help:

sudo -u www-data php occ -h

Display your ownCloud version:

```
sudo -u www-data php occ -V
ownCloud version 9.0.0
```

Query your ownCloud server status:

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

- installed: true
- version: 9.0.0.19
- versionstring: 9.0.0
- edition:

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

occ [options] command [arguments]

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

```
sudo -u www-data php occ help maintenance:mode
Usage:
maintenance:mode [options]
Options:
                       enable maintenance mode
     --on
     --off
                       disable maintenance mode
 -h, --help
                      Display this help message
 -q, --quiet Do not output any message
-V, --version Display this application version
     --ansi
                      Force ANSI output
     --no-ansi
                        Disable ANSI output
 -n, --no-interaction Do not ask any interactive question
     --no-warnings
                        Skip global warnings, show command output only
```

-v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

The status command from above has an option to define the output format. The default is plain text, but it can also be json:

```
sudo -u www-data php occ status --output=json
{"installed":true,"version":"9.0.0.19","versionstring":"9.0.0","edition":""}
```

or json_pretty:

```
sudo -u www-data php occ status --output=json_pretty
{
    "installed": true,
    "version": "9.0.0.19",
    "versionstring": "9.0.0",
    "edition": ""
}
```

This output option is available on all list and list-like commands: status, check, app:list, config:list, encryption:status and encryption:list-modules

5.3.3 Apps Commands

The app commands list, enable, and disable apps:

```
app

app:check-code check code to be compliant

app:disable disable an app

app:enable enable an app

app:getpath Get an absolute path to the app directory

(added in 9.0)

app:list List all available apps
```

List all of your installed apps, and show whether they are enabled or disabled:

sudo -u www-data php occ app:list

Enable an app, for example the External Storage Support app:

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

Disable an app:

sudo -u www-data php occ app:disable files_external
files_external disabled

app:check-code has multiple checks: it checks if an app uses ownCloud's public API (OCP) or private API (OC_), and it also checks for deprecated methods and the validity of the info.xml file. By default all checks are enabled. The Activity app is an example of a correctly-formatted app:

sudo -u www-data php occ app:check-code notifications
App is compliant - awesome job!

If your app has issues, you'll see output like this:

sudo -u www-data php occ app:check-code foo_app Analysing /var/www/owncloud/apps/files/foo_app.php 4 errors line 45: OCP\Response - Static method of deprecated class must not be called line 46: OCP\Response - Static method of deprecated class must not be called line 47: OCP\Response - Static method of deprecated class must not be called line 49: OC_Util - Static method of private class must not be called

You can get the full filepath to an app:

```
sudo -u www-data php occ app:getpath notifications
/var/www/owncloud/apps/notifications
```

5.3.4 Background Jobs Selector

Use the background command to select which scheduler you want to use for controlling background jobs, Ajax, Webcron, or Cron. This is the same as using the **Cron** section on your ownCloud Admin page:

```
background
background:ajax Use ajax to run background jobs
background:cron Use cron to run background jobs
background:webcron Use webcron to run background jobs
```

This example selects Ajax:

```
sudo -u www-data php occ background:ajax
  Set mode for background jobs to 'ajax'
```

The other two commands are:

- background:cron
- background:webcron

See Background Jobs to learn more.

5.3.5 Config Commands

The config commands are used to configure the ownCloud server:

```
configconfig:app:deleteDelete an app config valueconfig:app:getGet an app config valueconfig:app:setSet an app config valueconfig:importImport a list of configsconfig:listList all configsconfig:system:deleteDelete a system config valueconfig:system:getGet a system config valueconfig:system:setSet a system config value
```

You can list all configuration values with one command:

```
sudo -u www-data php occ config:list
```
By default, passwords and other sensitive data are omitted from the report, so the output can be posted publicly (e.g. as part of a bug report). In order to generate a full backport of all configuration values the --private flag needs to be set:

sudo -u www-data php occ config:list --private

The exported content can also be imported again to allow the fast setup of similar instances. The import command will only add or update values. Values that exist in the current configuration, but not in the one that is being imported are left untouched:

sudo -u www-data php occ config:import filename.json

It is also possible to import remote files, by piping the input:

sudo -u www-data php occ config:import < local-backup.json</pre>

Note: While it is possible to update/set/delete the versions and installation statuses of apps and ownCloud itself, it is not recommended to do this directly. Use the occ app:enable, occ app:disable and occ update commands instead.

Getting a Single Configuration Value

These commands get the value of a single app or system configuration:

```
sudo -u www-data php occ config:system:get version
9.0.0.19
```

```
sudo -u www-data php occ config:app:get activity installed_version 2.2.1 \,
```

Setting a Single Configuration Value

These commands set the value of a single app or system configuration:

```
sudo -u www-data php occ config:system:set logtimezone
--value="Europe/Berlin"
System config value logtimezone set to Europe/Berlin
```

```
sudo -u www-data php occ config:app:set files_sharing
incoming_server2server_share_enabled --value="yes" --type=boolean
Config value incoming_server2server_share_enabled for app files_sharing set to yes
```

The config:system:set command creates the value, if it does not already exist. To update an existing value, set --update-only:

```
sudo -u www-data php occ config:system:set doesnotexist --value="true"
--type=boolean --update-only
Value not updated, as it has not been set before.
```

Note that in order to write a Boolean, float, or integer value to the configuration file, you need to specify the type on your command. This applies only to the config:system:set command. The following values are known:

- boolean
- integer
- float

• string (default)

When you want to e.g. disable the maintenance mode run the following command:

```
sudo -u www-data php occ config:system:set maintenance --value=false
--type=boolean
ownCloud is in maintenance mode - no app have been loaded
System config value maintenance set to boolean false
```

Setting an array Configuration Value

Some configurations (e.g. the trusted domain setting) are an array of data. In order to set (and also get) the value of one key, you can specify multiple config names separated by spaces:

```
sudo -u www-data php occ config:system:get trusted_domains
localhost
owncloud.local
sample.tld
```

To replace sample.tld with example.com trusted_domains => 2 needs to be set:

```
sudo -u www-data php occ config:system:set trusted_domains 2
--value=example.com
System config value trusted_domains => 2 set to string example.com
```

sudo -u www-data php occ config:system:get trusted_domains
localhost
owncloud.local
example.com

Deleting a Single Configuration Value

These commands delete the configuration of an app or system configuration:

```
sudo -u www-data php occ config:system:delete maintenance:mode
System config value maintenance:mode deleted
```

sudo -u www-data php occ config:app:delete appname provisioning_api Config value provisioning_api of app appname deleted

The delete command will by default not complain if the configuration was not set before. If you want to be notified in that case, set the --error-if-not-exists flag:

```
sudo -u www-data php occ config:system:delete doesnotexist
--error-if-not-exists
Config provisioning_api of app appname could not be deleted because it did not
exist
```

5.3.6 Dav Commands

A set of commands to create addressbooks, calendars, and to migrate addressbooks from 8.2 when you upgrade to 9.0:

```
dav
dav:create-addressbook
dav:create-calendar
dav:migrate-addressbooks
Migrate addressbooks from the contacts
```

| | app to core |
|-----------------------------|--|
| dav:migrate-calendars | Migrate calendars from the calendar app to |
| | core |
| dav:sync-birthday-calendar | Synchronizes the birthday calendar |
| dav:sync-system-addressbook | Synchronizes users to the system |
| | addressbook |

The syntax for dav:create-addressbook and dav:create-calendar is dav:create-addressbook [user] [name]. This example creates the addressbook mollybook for the user molly:

sudo -u www-data php occ dav:create-addressbook molly mollybook

This example creates a new calendar for molly:

sudo -u www-data php occ dav:create-calendar molly mollycal

Molly will immediately see these on her Calendar and Contacts pages.

In 9.0, the CalDAV server has been integrated into core. Your existing calendars and contacts should migrate automatically when you upgrade. If something goes wrong you can try a manual migration. First delete any partially-migrated calendars or addressbooks. Then run this command to migrate user's contacts:

sudo -u www-data php occ dav:migrate-addressbooks [user]

Run this command to migrate calendars:

sudo -u www-data php occ dav:migrate-calendars [user]

See ownCloud 9.0 - calendar migration analysis for help with troubleshooting and reporting problems.

dav:sync-birthday-calendar adds all birthdays to your calendar from addressbooks shared with you. This example syncs to your calendar from user bernie:

sudo -u www-data php occ dav:sync-birthday-calendar bernie

dav:sync-system-addressbook synchronizes all users to the system addressbook:

```
sudo -u www-data php occ dav:sync-system-addressbook
```

Added in 9.0.

5.3.7 Database Conversion

The SQLite database is good for testing, and for ownCloud servers with small single-user workloads that do not use sync clients, but production servers with multiple users should use MariaDB, MySQL, or PostgreSQL. You can use occ to convert from SQLite to one of these other databases.

```
db
db:convert-type Convert the ownCloud database to the newly
configured one
db:generate-change-script generates the change script from the current
connected db to db_structure.xml
```

You need:

- Your desired database and its PHP connector installed.
- The login and password of a database admin user.
- The database port number, if it is a non-standard port.

This is example converts SQLite to MySQL/MariaDB:

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

For a more detailed explanation see Converting Database Type

5.3.8 Encryption

occ includes a complete set of commands for managing encryption:

| encryption | |
|------------------------------------|--|
| encryption:change-key-storage-root | Change key storage root |
| encryption:decrypt-all | Disable server-side encryption and |
| | decrypt all files |
| encryption:disable | Disable encryption |
| encryption:enable | Enable encryption |
| encryption:enable-master-key | Enable the master key. Only available |
| | for fresh installations with no existing |
| | encrypted data! There is also no way to |
| | disable it again. |
| encryption:encrypt-all | Encrypt all files for all users |
| encryption:list-modules | List all available encryption modules |
| encryption:migrate | initial migration to encryption 2.0 |
| encryption:set-default-module | Set the encryption default module |
| encryption:show-key-storage-root | Show current key storage root |
| encryption:status | Lists the current status of encryption |
| | |

encryption:status shows whether you have active encryption, and your default encryption module. To enable encryption you must first enable the Encryption app, and then run encryption:enable:

```
sudo -u www-data php occ app:enable encryption
sudo -u www-data php occ encryption:enable
sudo -u www-data php occ encryption:status
- enabled: true
- defaultModule: OC_DEFAULT_MODULE
```

encryption:change-key-storage-root is for moving your encryption keys to a different folder. It takes one argument, newRoot, which defines your new root folder. The folder must exist, and the path is relative to your root ownCloud directory:

sudo -u www-data php occ encryption:change-key-storage-root ../../etc/oc-keys

You can see the current location of your keys folder:

sudo -u www-data php occ encryption:show-key-storage-root Current key storage root: default storage location (data/)

encryption:list-modules displays your available encryption modules. You will see a list of modules only if you have enabled the Encryption app. Use encryption:set-default-module [module name] to set your desired module.

encryption:encrypt-all encrypts all data files for all users. You must first put your ownCloud server into *single-user mode* to prevent any user activity until encryption is completed.

encryption:decrypt-all decrypts all user data files, or optionally a single user:

sudo -u www-data php occ encryption:decrypt freda

Users must have enabled recovery keys on their Personal pages. You must first put your ownCloud server into *single-user mode* to prevent any user activity until decryption is completed.

Use encryption:disable to disable your encryption module. You must first put your ownCloud server into *single-user mode* to prevent any user activity.

encryption: enable-master-key creates a new master key, which is used for all user data instead of individual user keys. This is especially useful to enable single-sign on. Use this only on fresh installations with no existing data, or on systems where encryption has not already been enabled. It is not possible to disable it.

encryption:migrate migrates encryption keys after a major ownCloud version upgrade. You may optionally specify individual users in a space-delimited list.

See Encryption Configuration to learn more.

5.3.9 Federation Sync

Note: This command is only available when the "Federation" app (federation) is enabled.

Synchronize the addressbooks of all federated ownCloud servers:

federation:sync-addressbooks Synchronizes addressbooks of all federated clouds

In ownCloud 9.+, servers connected with federation shares can share user address books, and auto-complete usernames in share dialogs. Use this command to synchronize federated servers:

sudo -u www-data php occ federation:sync-addressbooks

Added in 9.0.

5.3.10 File Operations

occ has three commands for managing files in ownCloud:

| files | |
|--------------------------|---|
| files:cleanup | cleanup filecache |
| files:scan | rescan filesystem |
| files:transfer-ownership | All files and folders are moved to another |
| | user - shares are moved as well. (Added in 9.0) |

The files:scan command scans for new files and updates the file cache. You may rescan all files, per-user, a space-delimited list of users, and limit the search path. If not using --quiet, statistics will be shown at the end of the scan:

```
sudo -u www-data php occ files:scan --help
Usage:
files:scan [-p|--path="..."] [-q|--quiet] [-v|vv|vvv --verbose] [--all]
[user_idl] ... [user_idN]
Arguments:
user_id will rescan all files of the given user(s)
Options:
--path limit rescan to the user/path given
--all will rescan all files of all known users
--quiet suppress any output
```

| verbose | files | and | di | rectori | es being | processed | are | shown |
|---------|-------|-------|----|---------|----------|-----------|-----|-------|
| | addit | ional | ly | during | scanning | त्र | | |

Verbosity levels of -vv or -vvv are automatically reset to -v

When using the --path option, the path must consist of following components:

```
"user_id/files/path"
    or
"user_id/files/mount_name"
    or
"user_id/files/mount_name/path"
```

where the term files is mandatory.

Example:

--path="/alice/files/Music"

In the example above, the user_id alice is determined implicitly from the path component given.

The --path, --all and [user_id] parameters and are exclusive - only one must be specified.

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

You may transfer all files and shares from one user to another. This is useful before removing a user:

```
sudo -u www-data php occ files:transfer-ownership <source-user>
<destination-user>
```

5.3.11 Files External

These commands replace the data/mount.json configuration file used in ownCloud releases before 9.0.

Note: These commands are only available when the "External storage support" app (files_external) is enabled.

Commands for managing external storage:

```
files_external
files_external:applicable Manage applicable users and groups for a mount
files_external:backends Show available authentication and storage backends
files_external:config
                          Manage backend configuration for a mount
files_external:create
                        Create a new mount configuration
files_external:delete
                        Delete an external mount
files_external:export
                        Export mount configurations
files_external:import
                         Import mount configurations
files_external:list
                          List configured mounts
files_external:option
                          Manage mount options for a mount
files_external:verify
                          Verify mount configuration
```

These commands replicate the functionality in the ownCloud Web GUI, plus two new features: files_external:export and files_external:import.

Use files_external:export to export all admin mounts to stdout, and files_external:export [user_id] to export the mounts of the specified ownCloud user.

Use files_external:import [filename] to import legacy JSON configurations, and to copy external mount configurations to another ownCloud server.

Added in 9.0.

.

5.3.12 Integrity Check

Apps which have an official tag MUST be code signed starting with ownCloud 9.0. Unsigned official apps won't be installable anymore. Code signing is optional for all third-party applications:

| Integrity | |
|----------------------|--|
| integrity:check-app | Check app integrity using a signature. |
| integrity:check-core | Check core integrity using a signature |
| integrity:sign-app | Signs an app using a private key. |
| integrity:sign-core | Sign core using a private key |

After creating your signing key, sign your app like this example:

sudo -u www-data php occ integrity:sign-app --privateKey=/Users/lukasreschke/contacts.key --certifica

Verify your app:

sudo -u www-data php occ integrity:check-app --path=/pathto/app appname

When it returns nothing, your app is signed correctly. When it returns a message then there is an error. See Code Signing in the Developer manual for more detailed information.

integrity:sign-core is for ownCloud core developers only.

See Code Signing to learn more.

Added in 9.0.

5.3.13 I10n, Create Javascript Translation Files for Apps

This command is for app developers to update their translation mechanism from ownCloud 7 to ownCloud 8 and later.

5.3.14 LDAP Commands

Note: These commands are only available when the "LDAP user and group backend" app (user_ldap) is enabled.

These LDAP commands appear only when you have enabled the LDAP app. Then you can run the following LDAP commands with occ:

```
ldap
ldap:check-user
                             checks whether a user exists on LDAP.
ldap:create-empty-config
                             creates an empty LDAP configuration
ldap:delete-config
                             deletes an existing LDAP configuration
ldap:search
                             executes a user or group search
ldap:set-config
                             modifies an LDAP configuration
ldap:show-config
                             shows the LDAP configuration
ldap:show-remnants
                             shows which users are not available on
                             LDAP anymore, but have remnants in
                              ownCloud.
ldap:test-config
                             tests an LDAP configuration
```

Search for an LDAP user, using this syntax:

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

Searches will match at the beginning of the attribute value only. This example searches for givenNames that start with "rob":

sudo -u www-data php occ ldap:search "rob"

This will find robbie, roberta, and robin. Broaden the search to find, for example, jeroboam with the asterisk wildcard:

sudo -u www-data php occ ldap:search "*rob"

User search attributes are set with ldap:set-config (below). For example, if your search attributes are givenName and sn you can find users by first name + last name very quickly. For example, you'll find Terri Hanson by searching for te ha. Trailing whitespaces are ignored.

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

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

1dap:check-user will not run a check when it finds a disabled LDAP connection. This prevents users that exist on disabled LDAP connections from being marked as deleted. If you know for certain that the user you are searching for is not in one of the disabled connections, and exists on an active connection, use the --force option to force it to check all active LDAP connections:

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

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

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

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

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

Then you can list and view your configurations:

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

And view the configuration for a single configID:

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

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

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

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

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

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

sudo -u www-data php occ ldap:test-config s01
The configuration is valid and the connection could be established!

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

5.3.15 Logging Commands

These commands view and configure your ownCloud logging preferences:

```
log
log:manage manage logging configuration
log:owncloud manipulate ownCloud logging backend
```

Run log: owncloud to see your current logging status:

```
sudo -u www-data php occ log:owncloud
Log backend ownCloud: enabled
Log file: /opt/owncloud/data/owncloud.log
Rotate at: disabled
```

Use the --enable option to turn on logging. Use --file to set a different log file path. Set your rotation by log file size in bytes with --rotate-size; 0 disables rotation.

log:manage sets your logging backend, log level, and timezone. The defaults are owncloud, Warning, and UTC. Available options are:

- -backend [owncloud, syslog, errorlog]
- -level [debug, info, warning, error]

5.3.16 Maintenance Commands

Use these commands when you upgrade ownCloud, manage encryption, perform backups and other tasks that require locking users out until you are finished:

```
maintenance
maintenance:mimetype:update-db
maintenance:mimetype:update-js
maintenance:mode
maintenance:repair
maintenance:singleuser
Update database mimetypes and update
filecache
Update mimetypelist.js
set maintenance mode
repair this installation
set single user mode
```

maintenance:mode locks the sessions of all logged-in users, including administrators, and displays a status screen warning that the server is in maintenance mode. Users who are not already logged in cannot log in until maintenance mode is turned off. When you take the server out of maintenance mode logged-in users must refresh their Web browsers to continue working:

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

Putting your ownCloud server into single-user mode allows admins to log in and work, but not ordinary users. This is useful for performing maintenance and troubleshooting on a running server:

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

Turn it off when you're finished:

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

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

sudo -u www-data php occ maintenance:repair

maintenance:mimetype:update-db updates the ownCloud database and file cache with changed mimetypes found in config/mimetypemapping.json. Run this command after modifying config/mimetypemapping.json. If you change a mimetype, run maintenance:mimetype:update-db --repair-filecache to apply the change to existing files.

5.3.17 Security (Import SSL Certificates)

Use these commands to manage server-wide SSL certificates. These are useful when you create federation shares with other ownCloud servers that use self-signed certificates:

```
security
security:certificates list trusted certificates
security:certificates:import trusted certificate
security:certificates:remove remove trusted certificate
```

This example lists your installed certificates:

sudo -u www-data php occ security:certificates

Import a new certificate:

sudo -u www-data php occ security:import /path/to/certificate

Remove a certificate:

sudo -u www-data php occ security:remove [certificate name]

5.3.18 Shibboleth Modes (Enterprise Edition only)

Note: This command is only available when the "Shibboleth user backend" app (user_shibboleth) is enabled.

shibboleth:mode sets your Shibboleth mode to notactive, autoprovision, or ssoonly:

shibboleth:mode [mode]

5.3.19 Trashbin

Note: This command is only available when the "Deleted files" app (files_trashbin) is enabled.

The trashbin:cleanup command removes the deleted files of the specified users in a space-delimited list, or all users if none are specified.

```
trashbin
trashbin:cleanup Remove deleted files
```

This example removes the deleted files of all users:

```
sudo -u www-data php occ trashbin:cleanup
Remove all deleted files
Remove deleted files for users on backend Database
freda
molly
stash
rosa
edward
```

This example removes the deleted files of users molly and freda:

```
sudo -u www-data php occ trashbin:cleanup molly freda
Remove deleted files of molly
Remove deleted files of freda
```

5.3.20 User Commands

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

```
user
user:add adds a user
user:delete deletes the specified user
user:lastseen shows when the user was logged it last
time
user:report shows how many users have access
user:resetpassword Resets the password of the named user
```

You can create a new user with their display name, login name, and any group memberships with the user:add command. The syntax is:

```
user:add [--password-from-env] [--display-name[="..."]] [-g|--group[="..."]]
uid
```

The display-name corresponds to the **Full Name** on the Users page in your ownCloud Web UI, and the uid is their **Username**, which is their login name. This example adds new user Layla Smith, and adds her to the **users** and **db-admins** groups. Any groups that do not exist are created:

```
sudo -u www-data php occ user:add --display-name="Layla Smith"
--group="users" --group="db-admins" layla
Enter password:
Confirm password:
The user "layla" was created successfully
Display name set to "Layla Smith"
User "layla" added to group "users"
User "layla" added to group "db-admins"
```

Go to your Users page, and you will see your new user.

password-from-env allows you to set the user's password from an environment variable. This prevents the password from being exposed to all users via the process list, and will only be visible in the history of the user (root) running the command. This also permits creating scripts for adding multiple new users.

To use password-from-env you must run as "real" root, rather than sudo, because sudo strips environment variables. This example adds new user Fred Jones:

export OC_PASS=newpassword su -s /bin/sh www-data -c 'php occ user:add --password-from-env --display-name="Fred Jones" --group="users" fred' The user "fred" was created successfully Display name set to "Fred Jones" User "fred" added to group "users"

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

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

You may also use password-from-env to reset passwords:

```
export OC_PASS=newpassword
su -s /bin/sh www-data -c 'php occ user:resetpassword --password-from-env
layla'
Successfully reset password for layla
```

You can delete users:

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

View a user's most recent login:

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

Generate a simple report that counts all users, including users on external user authentication servers such as LDAP:

```
sudo -u www-data php occ user:report
+----+
           | |
| User Report
+----+
| Database | 12 |
| LDAP
           | 86 |
           | total users | 98 |
           | user directories | 2 |
+----+
```

5.3.21 Versions

Note: This command is only available when the "Versions" app (files_versions) is enabled.

Use this command to delete file versions for specific users, or for all users when none are specified:

versions Delete versions

This example deletes all versions for all users:

sudo -u www-data php occ versions:cleanup Delete all versions Delete versions for users on backend Database freda molly stash rosa edward

You can delete versions for specific users in a space-delimited list:

```
sudo -u www-data php occ versions:cleanup
Delete versions of freda
Delete versions of molly
```

5.3.22 Command Line Installation

These commands are available only after you have downloaded and unpacked the ownCloud archive, and taken no further installation steps.

You can install ownCloud entirely from the command line. After downloading the tarball and copying ownCloud into the appropriate directories, or after installing ownCloud packages (See *Preferred Linux Installation Method* and *Manual Installation on Linux*) you can use occ commands in place of running the graphical Installation Wizard.

Apply correct permissions to your ownCloud directories; see *Setting Strong Directory Permissions*. Then choose your occ options. This lists your available options:

```
sudo -u www-data php /var/www/owncloud/occ
ownCloud is not installed - only a limited number of commands are available
ownCloud version 9.0.0
Usage:
 [options] command [arguments]
Options:
--help (-h)
                      Display this help message
--quiet (-q)
                     Do not output any message
--verbose (-v|vv|vvv) Increase the verbosity of messages: 1 for normal
output, 2 for more verbose output and 3 for debug
--version (-V) Display this application version
--ansi
                     Force ANSI output
                     Disable ANSI output
 --no-ansi
 --no-interaction (-n) Do not ask any interactive question
Available commands:
check
                      check dependencies of the server environment
                      Displays help for a command
help
list
                      Lists commands
status
                      show some status information
app
app:check-code
                   check code to be compliant
110n
                      Create javascript translation files for a given app
l10n:createjs
maintenance
maintenance:install
                    install ownCloud
```

Display your maintenance: install options:

sudo -u www-data php occ help maintenance:install
ownCloud is not installed - only a limited number of commands are available
Usage:

```
maintenance:install [--database="..."] [--database-name="..."]
[--database-host="..."] [--database-user="..."] [--database-pass[="..."]]
[--database-table-prefix[="..."]] [--admin-user="..."] [--admin-pass="..."]
[--data-dir="..."]
Options:
 --database
                        Supported database type (default: "sqlite")
--database-name
                        Name of the database
--database-host
                       Hostname of the database (default: "localhost")
--database-user
                        User name to connect to the database
--database-pass
                       Password of the database user
--database-table-prefix Prefix for all tables (default: oc_)
 --admin-user
                       User name of the admin account (default: "admin")
 --admin-pass
                       Password of the admin account
--data-dir
                       Path to data directory (default:
                        "/var/www/owncloud/data")
                       Display this help message
--help (-h)
 --quiet (-q)
                        Do not output any message
--verbose (-v|vv|vvv) Increase the verbosity of messages: 1 for normal
 output, 2 for more verbose output and 3 for debug
                  Display this application version
 --version (-V)
                        Force ANSI output
--ansi
                       Disable ANSI output
 --no-ansi
 --no-interaction (-n) Do not ask any interactive question
```

This example completes the installation:

```
cd /var/www/owncloud/
sudo -u www-data php occ maintenance:install --database
"mysql" --database-name "owncloud" --database-user "root" --database-pass
"password" --admin-user "admin" --admin-pass "password"
ownCloud is not installed - only a limited number of commands are available
ownCloud was successfully installed
```

Supported databases are:

```
sqlite (SQLite3 - ownCloud Community edition only)
mysql (MySQL/MariaDB)
pgsql (PostgreSQL)
oci (Oracle - ownCloud Enterprise edition only)
```

5.3.23 Command Line Upgrade

These commands are available only after you have downloaded upgraded packages or tar archives, and before you complete the upgrade.

List all options, like this example on CentOS Linux:

```
sudo -u apache php occ upgrade -h
Usage:
upgrade [--skip-migration-test] [--dry-run] [--no-app-disable]
Options:
--skip-migration-test skips the database schema migration simulation and
update directly
--dry-run only runs the database schema migration simulation, do
not actually update
--no-app-disable skips the disable of third party apps
```

```
--help (-h) Display this help message.
--quiet (-q) Do not output any message.
--verbose (-v|vv|vvv) Increase the verbosity of messages: 1 for normal output,
2 for more verbose output and 3 for debug.
--version (-V) Display this application version.
--ansi Force ANSI output.
--no-ansi Disable ANSI output.
--no-interaction (-n) Do not ask any interactive question
```

When you are performing an update or upgrade on your ownCloud server (see the Maintenance section of this manual), it is better to use occ to perform the database upgrade step, rather than the Web GUI, in order to avoid timeouts. PHP scripts invoked from the Web interface are limited to 3600 seconds. In larger environments this may not be enough, leaving the system in an inconsistent state. After performing all the preliminary steps (see *How to Upgrade Your ownCloud Server*) use this command to upgrade your databases, like this example on CentOS Linux. Note how it details the steps:

```
sudo -u www-data php occ upgrade
ownCloud or one of the apps require upgrade - only a limited number of
commands are available
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Updating <gallery> ...
Updated <gallery> to 0.6.1
Updating <activity> ...
Updated <activity> to 2.1.0
Update successful
Turned off maintenance mode
```

Enabling verbosity displays timestamps:

sudo -u www-data php occ upgrade -v
ownCloud or one of the apps require upgrade - only a limited number of commands are available
2015-06-23T09:06:15+0000 Turned on maintenance mode
2015-06-23T09:06:15+0000 Checked database schema update
2015-06-23T09:06:15+0000 Updated database
2015-06-23T09:06:15+0000 Updated <files_sharing> to 0.6.6
2015-06-23T09:06:15+0000 Update successful
2015-06-23T09:06:15+0000 Turned off maintenance mode

If there is an error it throws an exception, and the error is detailed in your ownCloud logfile, so you can use the log output to figure out what went wrong, or to use in a bug report:

Turned on maintenance mode Checked database schema update Checked database schema update for apps Updated database Updating <files_sharing> ... Exception ServerNotAvailableException: LDAP server is not available Update failed Turned off maintenance mode

Before completing the upgrade, ownCloud first runs a simulation by copying all database tables to new tables, and then performs the upgrade on them, to ensure that the upgrade will complete correctly. The copied tables are deleted after the upgrade. This takes twice as much time, which on large installations can be many hours, so you can omit this step with the --skip-migration-test option:

sudo -u www-data php occ upgrade --skip-migration-test

You can perform this simulation manually with the --dry-run option:

sudo -u www-data php occ upgrade --dry-run

5.4 Configuring the Activity App

You can configure your ownCloud server to automatically send out e-mail notifications to your users for various events like:

- A file or folder has been shared
- A new file or folder has been created
- A file or folder has been changed
- A file or folder has been deleted

Users can see actions (delete, add, modify) that happen to files they have access to. Sharing actions are only visible to the sharer and sharee.

5.4.1 Enabling the Activity App

The Activity App is shipped and enabled by default. If it is not enabled simply go to your ownCloud Apps page to enable it.

5.4.2 Configuring your ownCloud for the Activity App

To configure your ownCloud to send out e-mail notifications a working *Email Configuration* is mandatory.

Furthermore it is recommended to configure the background job Webcron or Cron as described in *Background Jobs*.

There is also a configuration option activity_expire_days available in your config.php (See *Config.php Parameters*) which allows you to clean-up older activies from the database.

5.5 Configuring the ClamAV Antivirus Scanner

You can configure your ownCloud server to automatically run a virus scan on newly-uploaded files with the Antivirus App for Files. The Antivirus App for Files integrates the open source anti-virus engine ClamAV with ownCloud. ClamAV detects all forms of malware including Trojan horses, viruses, and worms, and it operates on all major file types including Windows, Linux, and Mac files, compressed files, executables, image files, Flash, PDF, and many others. ClamAV's Freshclam daemon automatically updates its malware signature database at scheduled intervals.

ClamAV runs on Linux and any Unix-type operating system, and Microsoft Windows. However, it has only been tested with ownCloud on Linux, so these instructions are for Linux systems. You must first install ClamAV, and then install and configure the Antivirus App for Files on ownCloud.

5.5.1 Installing ClamAV

As always, the various Linux distributions manage installing and configuring ClamAV in different ways.

Debian, Ubuntu, Linux Mint On Debian and Ubuntu systems, and their many variants, install ClamAV with these commands:

apt-get install clamav clamav-daemon

The installer automatically creates default configuration files and launches the clamd and freshclam daemons. You don't have to do anything more, though it's a good idea to review the ClamAV documentation and your settings in /etc/clamav/. Enable verbose logging in both clamd.conf and freshclam.conf until you get any kinks worked out.

Red Hat 7, CentOS 7 On Red Hat 7 and related systems you must install the Extra Packages for Enterprise Linux (EPEL) repository, and then install ClamAV:

```
yum install epel-release
yum install clamav clamav-scanner clamav-scanner-systemd clamav-server
clamav-server-systemd clamav-update
```

This installs two configuration files: /etc/freshclam.conf and /etc/clamd.d/scan.conf. You must edit both of these before you can run ClamAV. Both files are well-commented, and man clamd.conf and man freshclam.conf explain all the options. Refer to /etc/passwd and /etc/group when you need to verify the ClamAV user and group.

First edit /etc/freshclam.conf and configure your options. freshclam updates your malware database, so you want it to run frequently to get updated malware signatures. Run it manually post-installation to download your first set of malware signatures:

freshclam

The EPEL packages do not include an init file for freshclam, so the quick and easy way to set it up for regular checks is with a cron job. This example runs it every hour at 47 minutes past the hour:

m h dom mon dow command 47 * * * * /usr/bin/freshclam --quiet

Please avoid any multiples of 10, because those are when the ClamAV servers are hit the hardest for updates.

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

```
systemctl enable clamd@scan.service systemctl start clamd@scan.service
```

That should take care of everything. Enable verbose logging in scan.conf and freshclam.conf until it is running the way you want.

5.5.2 Enabling the Antivirus App for Files

Simply go to your ownCloud Apps page to enable it.

5.5.3 Configuring ClamAV on ownCloud

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

Now find your Antivirus Configuration panel on your Admin page.

Antivirus App for files 0.4.2

Verify files for virus using ClamAV

<u>See application page at apps.owncloud.com</u> AGPL-licensed by Manuel Delgado, Bart Visscher, thinksilicon.de

Enable

| Log | | |
|-----------|--|---|
| Log level | Everything (fatal issues, errors, warnings, info, debug) | • |

| Antivir | us Configuration | | |
|---------|-----------------------------|--------------|------------|
| Mode | Daemon (Socket) 🔻 | | |
| Socket | /var/run/clamav/cla | | |
| Stream | Length 1024 | bytes | |
| Action | for infected files found wh | ile scanning | Only log 🔹 |
| Save |] | | |

ClamAV runs in one of three modes:

- Daemon (Socket): ClamAV is running on the same server as ownCloud. The ClamAV daemon, clamd, runs in the background. When there is no activity clamd places a minimal load on your system. If your users upload large volumes of files you will see high CPU usage.
- Daemon: ClamAV is running on a different server. This is a good option for ownCloud servers with high volumes of file uploads.
- Executable: ClamAV is running on the same server as ownCloud, and the clamscan command is started and then stopped with each file upload. clamscan is slow and not always reliable for on-demand usage; it is better to use one of the daemon modes.
- Daemon (Socket) ownCloud should detect your clamd socket and fill in the Socket field. This is the LocalSocket option in clamd.conf. You can run netstat to verify:

| netstat
unix 2 [| -a gr
ACC | rep
] | o clam
STREAM | LISTENING | 15857 | /var/run/clamav/clamd.ctl | | |
|---------------------|--------------|----------|------------------|-----------|---------|---------------------------|--|--|
| | | | Ant | ivi | rus Cor | nfiguration | | |

| Antivirus configuration | |
|---------------------------------------|-------------------------|
| Mode Daemon (Socket) | |
| Socket /var/run/clamav/cla | |
| Stream Length 1024 | bytes |
| Action for infected files found while | scanning Only log |
| Save | Only log
Delete file |

The Stream Length value sets the number of bytes read in one pass. 10485760 bytes, or ten megabytes, is the default. This value should be no larger than the PHP memory_limit settings, or physical memory if memory_limit is set to -1 (no limit).

Action for infected files found while scanning gives you the choice of logging any alerts without deleting the files, or immediately deleting infected files.

- **Daemon** For the Daemon option you need the hostname or IP address of the remote server running ClamAV, and the server's port number.
- **Executable** The Executable option requires the path to clamscan, which is the interactive ClamAV scanning command. ownCloud should find it automatically.

When you are satisfied with how ClamAV is operating, you might want to go back and change all of your logging to less verbose levels.

5.6 Configuring Memory Caching

You can significantly improve your ownCloud server performance with memory caching, where frequently-requested objects are stored in memory for faster retrieval. There are two types of caches to use: a PHP opcode cache, which is commonly called *opcache*, and data caching for your Web server. If you do not install and enable a local memcache you will see a warning on your ownCloud admin page. A memcache is not required and you may safely ignore the warning if you prefer.

| Antivirus Configuration | | | | | | | |
|-------------------------|-----------------------------|---------------|-------------------------|--|--|--|--|
| Mode | Daemon (Socket) 🔽 | | | | | | |
| Socket | /var/run/clamav/cla | | | | | | |
| Stream | Length 1024 | bytes | | | | | |
| Action | for infected files found wi | nile scanning | Only log 🔻 | | | | |
| Save | | | Only log
Delete file | | | | |

| Antivir | us Configu | ration | | | |
|---------|-------------|---------------------|----------|----------|---|
| Mode | Executab | le 🗾 | | | |
| Strean | n Length 1 | 0485760 | bytes | | |
| Path to | o clamscan | /usr/bin/clamscar | 1 | | |
| Action | for infecte | d files found while | scanning | Only log | • |
| Save |] | | | | |

Note: If you enable only a distributed cache in your config.php (memcache.distributed) and not a local cache (memcache.local) you will still see the cache warning.

A PHP opcache stores compiled PHP scripts so they don't need to be re-compiled every time they are called. PHP bundles the Zend OPcache in core since version 5.5, so you don't need to install an opcache for PHP 5.5+.

If you are using PHP 5.4, which is the oldest supported PHP version for ownCloud, you may install the Alternative PHP Cache (APC). This is both an opcache and data cache. APC has not been updated since 2012 and is essentially dead, and PHP 5.4 is old and lags behind later releases. If it is possible to upgrade to a later PHP release that is the best option.

Data caching is supplied by the Alternative PHP Cache, user (APCu) in PHP 5.5+, Memcached, or Redis.

ownCloud supports multiple memory caching backends, so you can choose the type of memcache that best fits your needs. The supported caching backends are:

- APC A local cache for systems running PHP 5.4.
- APCu, APCu 4.0.6 and up required. A local cache for systems running PHP 5.5 and up.
- Memcached Distributed cache for multi-server ownCloud installations.
- Redis, PHP module 2.2.6 and up required. For distributed caching.

Memcaches must be explicitly configured in ownCloud 8.1 and up by installing and enabling your desired cache, and then adding the appropriate entry to config.php (See *Config.php Parameters* for an overview of all possible config parameters).

You may use both a local and a distributed cache. Recommended caches are APCu and Redis. After installing and enabling your chosen memcache, verify that it is active by running *PHP Version and Information*.

5.6.1 APC

APC is only for systems running PHP 5.4 and older. The oldest supported PHP version in ownCloud is 5.4.

Note: RHEL 6 and CentOS 6 ship with PHP 5.3 and must be upgraded to PHP 5.4 to run ownCloud. See *Installing PHP 5.4 on RHEL 6 and CentOS 6*.

On Red Hat/CentOS/Fedora systems running PHP 5.4, install php-pecl-apc. On Debian/Ubuntu/Mint systems install php-apc. Then restart your Web server.

After restarting your Web server, add this line to your config.php file:

'memcache.local' => '\OC\Memcache\APC',

Refresh your ownCloud admin page, and the cache warning should disappear.

5.6.2 APCu

PHP 5.5 and up include the Zend OPcache in core, and on most Linux distributions it is enabled by default. However, it does not bundle a data cache. APCu is a data cache, and it is available in most Linux distributions. On Red Hat/CentOS/Fedora systems running PHP 5.5 and up install php-pecl-apcu. On Debian/Ubuntu/Mint systems install php5-apcu. On Ubuntu 14.04LTS, the APCu version is 4.0.2, which is too old to use with ownCloud. ownCloud requires 4.0.6+. You may install 4.0.7 from Ubuntu backports with this command:

apt-get install php5-apcu/trusty-backports

Then restart your Web server.

After restarting your Web server, add this line to your config.php file:

'memcache.local' => '\OC\Memcache\APCu',

Refresh your ownCloud admin page, and the cache warning should disappear.

5.6.3 Memcached

Memcached is a reliable oldtimer for shared caching on distributed servers, and performs well with ownCloud with one exception: it is not suitable to use with *Transactional File Locking* because it does not store locks, and data can disappear from the cache at any time (Redis is the best memcache for this).

Note: Be sure to install the **memcached** PHP module, and not memcache, as in the following examples. ownCloud supports only the **memcached** PHP module.

Setting up Memcached is easy. On Debian/Ubuntu/Mint install memcached and php5-memcached. The installer will automatically start memcached and configure it to launch at startup.

On Red Hat/CentOS/Fedora install memcached and php-pecl-memcached. It will not start automatically, so you must use your service manager to start memcached, and to launch it at boot as a daemon.

You can verify that the Memcached daemon is running with ps ax:

```
ps ax | grep memcached
19563 ? Sl 0:02 /usr/bin/memcached -m 64 -p 11211 -u memcache -l
127.0.0.1
```

Restart your Web server, add the appropriate entries to your config.php, and refresh your ownCloud admin page. This example uses APCu for the local cache, Memcached as the distributed memcache, and lists all the servers in the shared cache pool with their port numbers:

5.6.4 Redis

Redis is an excellent modern memcache to use for both distributed caching, and as a local cache for *Transactional File Locking* because it guarantees that cached objects are available for as long as they are needed.

The Redis PHP module must be version 2.2.6+. If you are running a Linux distribution that does not package the supported versions of this module, or does not package Redis at all, see *Additional Redis Installation Help*.

On Debian/Ubuntu/Mint install redis-server and php5-redis. The installer will automatically launch redis-server and configure it to launch at startup.

On CentOS and Fedora install redis and php-pecl-redis. It will not start automatically, so you must use your service manager to start redis, and to launch it at boot as a daemon.

You can verify that the Redis daemon is running with ps ax:

```
ps ax | grep redis
22203 ? Ssl 0:00 /usr/bin/redis-server 127.0.0.1:6379
```

Restart your Web server, add the appropriate entries to your config.php, and refresh your ownCloud admin page. This example config.php configuration uses Redis for the local server cache:

For best performance, use Redis for file locking by adding this:

```
'memcache.locking' => '\OC\Memcache\Redis',
```

If you want to connect to Redis configured to listen on an Unix socket (which is recommended if Redis is running on the same system as ownCloud) use this example config.php configuration:

Redis is very configurable; consult the Redis documentation to learn more.

5.6.5 Cache Directory Location

The cache directory defaults to data/\$user/cache where \$user is the current user. You may use the 'cache_path' directive in config.php (See *Config.php Parameters*) to select a different location.

5.6.6 Recommendations Based on Type of Deployment

Small/Private Home Server

Only use APCu:

```
'memcache.local' => '\OC\Memcache\APCu',
```

Small Organization, Single-server Setup

Use APCu for local caching, Redis for file locking:

Large Organization, Clustered Setup

Use Redis for everything except local memcache. Use the server's IP address or hostname so that it is accessible to other hosts:

Additional notes for Redis vs. APCu on Memory Caching

APCu is faster at local caching than Redis. If you have enough memory, use APCu for Memory Caching and Redis for File Locking. If you are low on memory, use Redis for both.

5.6.7 Additional Redis Installation Help

If your version of Mint or Ubuntu does not package the required version of php5-redis, then try this Redis guide on Tech and Me for a complete Redis installation on Ubuntu 14.04 using PECL. These instructions are adaptable for any distro that does not package the supported version, or that does not package Redis at all, such as SUSE Linux Enterprise Server and Red Hat Enterprise Linux.

The Redis PHP module must be at least version 2.2.6.

See https://pecl.php.net/package/redis

On Debian/Mint/Ubuntu, use apt-cache to see the available php5-redis version, or the version of your installed package:

apt-cache policy php5-redis

On CentOS and Fedora, the yum command shows available and installed version information:

yum search php-pecl-redis

5.7 Background Jobs

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

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

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

5.7.1 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

5.7.2 Cron Jobs

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

AJAX

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

Note: Especially when using the Activity App or external storages, where new files are added, updated or deleted one of the two methods below should be preferred.

Webcron

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

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

Cron

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

To run a cron job on a *nix system, every 15 minutes, under the default Web server user (often, www-data or wwwrun), you must set up the following cron job to call the **cron.php** script:

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

You can verify if the cron job has been added and scheduled by executing:

crontab -u www-data -l
*/15 * * * * php -f /var/www/owncloud/cron.php

Note: You have to replace the path /var/www/owncloud/cron.php with the path to your current ownCloud installation.

Note: On some systems it might be required to call php-cli instead of php.

Available Background Jobs

A number of existing background jobs are available to be run just for specific tasks.

Note: These jobs are generally only needed on large instances and can be run as background jobs. If the number of users in your installation ranges between 1,000 and 3,000, or if you're using LDAP and it becomes a bottleneck, then admins can delete several entries in the *oc_jobs* table and replace them with the corresponding *occ* command, which you can see here:

- OCA\Files_Trashbin\BackgroundJob\ExpireTrash -> occ trashbin:expire
- OCA\Files_Versions\BackgroundJob\ExpireVersions -> occ versions:expire
- OCA\DAVCardDAV\SyncJob -> occ dav:sync-system-addressbook
- OCA\Federation\SyncJob -> occ federation:sync-addressbooks

If used, these should be scheduled to run on a daily basis.

While not exhaustive, these include:

ExpireTrash

The ExpireTrash job, contained in OCA\Files_Trashbin\BackgroundJob\ExpireTrash, will remove any file in the ownCloud trash bin which is older than the specified maximum file retention time. It can be run, as follows, using the OCC command:

occ trashbin:expire

ExpireVersions

The ExpireVersions job, contained in OCA\Files_Versions\BackgroundJob\ExpireVersions, will expire versions of files which are older than the specified maximum version retention time. It can be run, as follows, using the OCC command:

occ versions:expire

SyncJob (CardDAV)

The CardDAV SyncJob, contained in OCA\DAV\CardDAV\SyncJob, syncs the local system address book, updating any existing contacts, and deleting any expired contacts. It can be run, as follows, using the OCC command:

occ dav:sync-system-addressbook

SyncJob (Federation)

OCAFederationSyncJob

It can be run, as follows, using the OCC command:

occ federation:sync-addressbooks

5.8 Config.php Parameters

ownCloud uses the config/config.php file to control server operations. config/config.sample.php lists all the configurable parameters within ownCloud, along with example or default values. This document provides a more detailed reference. Most options are configurable on your Admin page, so it is usually not necessary to edit config/config.php.

Note: The installer creates a configuration containing the essential parameters. Only manually add configuration parameters to config/config.php if you need to use a special value for a parameter. **Do not copy everything** from config/config.sample.php. **Only enter the parameters you wish to modify!**

ownCloud supports loading configuration parameters from multiple files. You can add arbitrary files ending with .config.php in the config/ directory, for example you could place your email server configuration in email.config.php. This allows you to easily create and manage custom configurations, or to divide a large complex configuration file into a set of smaller files. These custom files are not overwritten by ownCloud, and the values in these files take precedence over config.php.

5.8.1 Default Parameters

These parameters are configured by the ownCloud installer, and are required for your ownCloud server to operate.

```
'instanceid' => '',
```

This is a unique identifier for your ownCloud installation, created automatically by the installer. This example is for documentation only, and you should never use it because it will not work. A valid instanceid is created when you install ownCloud.

```
'instanceid' => 'd3c944a9a',
'passwordsalt' => '',
```

The salt used to hash all passwords, auto-generated by the ownCloud installer. (There are also per-user salts.) If you lose this salt you lose all your passwords. This example is for documentation only, and you should never use it.

```
'trusted_domains' =>
  array (
        'demo.example.org',
        'otherdomain.example.org',
    ),
```

Your list of trusted domains that users can log into. Specifying trusted domains prevents host header poisoning. Do not remove this, as it performs necessary security checks.

```
'datadirectory' => '/var/www/owncloud/data',
```

Where user files are stored; this defaults to data/ in the ownCloud directory. The SQLite database is also stored here, when you use SQLite.

(SQLite is not available in ownCloud Enterprise Edition)

'version' => '',

The current version number of your ownCloud installation. This is set up during installation and update, so you shouldn't need to change it.

'dbtype' => 'sqlite',

Identifies the database used with this installation. See also config option supportedDatabases

Available:

- sqlite (SQLite3 Not in Enterprise Edition)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle Enterprise Edition Only)

'dbhost' => '',

Your host server name, for example localhost, hostname, hostname.example.com, or the IP address. To specify a port use hostname:####; to specify a Unix socket use localhost:/path/to/socket.

```
'dbname' => 'owncloud',
```

The name of the ownCloud database, which is set during installation. You should not need to change this.

'dbuser' => '',

The user that ownCloud uses to write to the database. This must be unique across ownCloud instances using the same SQL database. This is set up during installation, so you shouldn't need to change it.

'dbpassword' => '',

The password for the database user. This is set up during installation, so you shouldn't need to change it.

```
'dbtableprefix' => '',
```

Prefix for the ownCloud tables in the database.

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

Indicates whether the ownCloud instance was installed successfully; true indicates a successful installation, and false indicates an unsuccessful installation.

5.8.2 Default config.php Examples

When you use SQLite as your ownCloud database, your config.php looks like this after installation. The SQLite database is stored in your ownCloud data/ directory. SQLite is a simple, lightweight embedded database that is good for testing and for simple installations, but for production ownCloud systems you should use MySQL, MariaDB, or PosgreSQL.

```
<?php
$CONFIG = array (
    'instanceid' => 'occ6f7365735',
    'passwordsalt' => '2c5778476346786306303',
    'trusted_domains' =>
    array (
        0 => 'localhost',
        1 => 'studio',
        1 => 'studio',
```

```
),
 'datadirectory' => '/var/www/owncloud/data',
 'dbtype' => 'sqlite3',
 'version' => '7.0.2.1',
 'installed' => true,
);
```

This example is from a new ownCloud installation using MariaDB:

```
<?php
CONFIG = array (
  'instanceid' => 'oc8c0fd71e03',
  'passwordsalt' => '515a13302a6b3950a9d0fdb970191a',
  'trusted_domains' =>
  array (
   0 => 'localhost',
    1 => 'studio',
    2 => '192.168.10.155'
  ),
  'datadirectory' => '/var/www/owncloud/data',
  'dbtype' => 'mysql',
  'version' => '7.0.2.1',
  'dbname' => 'owncloud',
  'dbhost' => 'localhost',
  'dbtableprefix' => 'oc_',
  'dbuser' => 'oc_carla',
  'dbpassword' => '67336bcdf7630dd80b2b81a413d07',
  'installed' => true,
);
```

5.8.3 User Experience

These optional parameters control some aspects of the user interface. Default values, where present, are shown.

'default_language' => 'en',

This sets the default language on your ownCloud server, using ISO_639-1 language codes such as en for English, de for German, and fr for French. It overrides automatic language detection on public pages like login or shared items. User's language preferences configured under "personal -> language" override this setting after they have logged in.

```
'defaultapp' => 'files',
```

Set the default app to open on login. Use the app names as they appear in the URL after clicking them in the Apps menu, such as documents, calendar, and gallery. You can use a comma-separated list of app names, so if the first app is not enabled for a user then ownCloud will try the second one, and so on. If no enabled apps are found it defaults to the Files app.

```
'knowledgebaseenabled' => true,
```

true enables the Help menu item in the user menu (top right of the ownCloud Web interface). false removes the Help item.

'enable_avatars' => true,

true enables avatars, or user profile photos. These appear on the User page, on user's Personal pages and are used by some apps (contacts, mail, etc). false disables them.

'allow_user_to_change_display_name' => true,

true allows users to change their display names (on their Personal pages), and false prevents them from changing their display names.

```
'remember_login_cookie_lifetime' => 60*60*24*15,
```

Lifetime of the remember login cookie, which is set when the user clicks the remember checkbox on the login screen. The default is 15 days, expressed in seconds.

'session_lifetime' => 60 * 60 * 24,

The lifetime of a session after inactivity; the default is 24 hours, expressed in seconds.

'session_keepalive' => true,

Enable or disable session keep-alive when a user is logged in to the Web UI.

Enabling this sends a "heartbeat" to the server to keep it from timing out.

'skeletondirectory' => '/path/to/owncloud/core/skeleton',

The directory where the skeleton files are located. These files will be copied to the data directory of new users. Leave empty to not copy any skeleton files.

The user_backends app (which needs to be enabled first) allows you to configure alternate authentication backends. Supported backends are: IMAP (OC_User_IMAP), SMB (OC_User_SMB), and FTP (OC_User_FTP).

'lost_password_link' => 'https://example.org/link/to/password/reset',

If your user backend does not allow to reset the password (e.g. when it's a read-only user backend like LDAP), you can specify a custom link, where the user is redirected to, when clicking the "reset password" link after a failed login-attempt.

5.8.4 Mail Parameters

These configure the email settings for ownCloud notifications and password resets.

'mail_domain' => 'example.com',

The return address that you want to appear on emails sent by the ownCloud server, for example oc-admin@example.com, substituting your own domain, of course.

'mail_from_address' => 'owncloud',

FROM address that overrides the built-in sharing-noreply and lostpassword-noreply FROM addresses.

'mail_smtpdebug' => false,

Enable SMTP class debugging.

'mail_smtpmode' => 'sendmail',

Which mode to use for sending mail: sendmail, smtp, qmail or php.

If you are using local or remote SMTP, set this to smtp.

If you are using PHP mail you must have an installed and working email system on the server. The program used to send email is defined in the php.ini file.

For the sendmail option you need an installed and working email system on the server, with /usr/sbin/sendmail installed on your Unix system.

For qmail the binary is /var/qmail/bin/sendmail, and it must be installed on your Unix system.

'mail_smtphost' => '127.0.0.1',

This depends on mail_smtpmode. Specify the IP address of your mail server host. This may contain multiple hosts separated by a semi-colon. If you need to specify the port number append it to the IP address separated by a colon, like this: 127.0.0.1:24.

'mail_smtpport' => 25,

This depends on mail_smtpmode. Specify the port for sending mail.

```
'mail_smtptimeout' => 10,
```

This depends on mail_smtpmode. This sets the SMTP server timeout, in seconds. You may need to increase this if you are running an anti-malware or spam scanner.

'mail_smtpsecure' => '',

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

'mail_smtpauth' => false,

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

```
'mail_smtpauthtype' => 'LOGIN',
```

This depends on mail_smtpmode. If SMTP authentication is required, choose the authentication type as LOGIN (default) or PLAIN.

'mail_smtpname' => '',

This depends on mail_smtpauth. Specify the username for authenticating to the SMTP server.

'mail_smtppassword' => '',

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

5.8.5 Proxy Configurations

```
'overwritehost' => '',
```

The automatic hostname detection of ownCloud can fail in certain reverse proxy and CLI/cron situations. This option allows you to manually override the automatic detection; for example.com, or specify the port www.example.com:8080.

'overwriteprotocol' => '',

When generating URLs, ownCloud attempts to detect whether the server is accessed via https or http. However, if ownCloud is behind a proxy and the proxy handles the https calls, ownCloud would not know that ssl is in use, which would result in incorrect URLs being generated.

Valid values are http and https.

'overwritewebroot' => '',

ownCloud attempts to detect the webroot for generating URLs automatically.

For example, if www.example.com/owncloud is the URL pointing to the ownCloud instance, the webroot is /owncloud. When proxies are in use, it may be difficult for ownCloud to detect this parameter, resulting in invalid URLs.

```
'overwritecondaddr' => '',
```

```
'overwrite.cli.url' => '',
```

Use this configuration parameter to specify the base URL for any URLs which are generated within own-Cloud using any kind of command line tools (cron or occ). The value should contain the full base URL: https://www.example.com/owncloud

'htaccess.RewriteBase' => '/',

To have clean URLs without /index.php this parameter needs to be configured.

This parameter will be written as "RewriteBase" on update and installation of ownCloud to your *.htaccess* file. While this value is often simply the URL path of the ownCloud installation it cannot be set automatically properly in every scenario and needs thus some manual configuration.

In a standard Apache setup this usually equals the folder that ownCloud is accessible at. So if ownCloud is accessible via "https://mycloud.org/owncloud" the correct value would most likely be "/owncloud". If ownCloud is running under "https://mycloud.org/" then it would be "/".

Note that above rule is not valid in every case, there are some rare setup cases where this may not apply. However, to avoid any update problems this configuration value is explicitly opt-in.

After setting this value run *occ maintenance:update:htaccess* and when following conditions are met ownCloud uses URLs without index.php in it:

- *mod_rewrite* is installed
- mod_env is installed

'proxy' => '',

The URL of your proxy server, for example proxy.example.com:8081.

```
'proxyuserpwd' => '',
```

The optional authentication for the proxy to use to connect to the internet.

The format is: username:password.

5.8.6 Deleted Items (trash bin)

These parameters control the Deleted files app.

```
'trashbin_retention_obligation' => 'auto',
```

If the trash bin app is enabled (default), this setting defines the policy for when files and folders in the trash bin will be permanently deleted.

The app allows for two settings, a minimum time for trash bin retention, and a maximum time for trash bin retention. Minimum time is the number of days a file will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define file and folder deletion. For migration purposes, this setting is installed initially set to "auto", which is equivalent to the default setting in ownCloud 8.1 and before.

Available values:

- **auto** default setting. keeps files and folders in the trash bin for 30 days and automatically deletes anytime after that if space is needed (note: files may not be deleted if space is not needed).
- D, auto keeps files and folders in the trash bin for D+ days, delete anytime if space needed (note: files may not be deleted if space is not needed)
- auto, D delete all files in the trash bin that are older than D days automatically, delete other files anytime if space needed
- D1, D2 keep files and folders in the trash bin for at least D1 days and delete when exceeds D2 days
- disabled trash bin auto clean disabled, files and folders will be kept forever

5.8.7 File versions

These parameters control the Versions app.

```
'versions_retention_obligation' => 'auto',
```

If the versions app is enabled (default), this setting defines the policy for when versions will be permanently deleted.

The app allows for two settings, a minimum time for version retention, and a maximum time for version retention. Minimum time is the number of days a version will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define version deletion. For migration purposes, this setting is installed initially set to "auto", which is equivalent to the default setting in ownCloud 8.1 and before.

Available values:

- **auto** default setting. Automatically expire versions according to expire rules. Please refer to Files_versions online documentation for more info.
- D, auto keep versions at least for D days, apply expire rules to all versions that are older than D days
- auto, D delete all versions that are older than D days automatically, delete other versions according to expire rules
- D1, D2 keep versions for at least D1 days and delete when exceeds D2 days
- disabled versions auto clean disabled, versions will be kept forever

5.8.8 ownCloud Verifications

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

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

Checks an app before install whether it uses private APIs instead of the proper public APIs. If this is set to true it will only allow to install or enable apps that pass this check.

'updatechecker' => true,

Check if ownCloud is up-to-date and shows a notification if a new version is available.

'updater.server.url' => 'https://updates.owncloud.com/server/',

URL that ownCloud should use to look for updates

'has_internet_connection' => true,

Is ownCloud connected to the Internet or running in a closed network?

'check_for_working_webdav' => true,

Allows ownCloud to verify a working WebDAV connection. This is done by attempting to make a WebDAV request from PHP.

'check_for_working_wellknown_setup' => true,

Allows ownCloud to verify a working .well-known URL redirects. This is done by attempting to make a request from JS to https://your-domain.com/.well-known/caldav/

'check_for_working_htaccess' => true,

This is a crucial security check on Apache servers that should always be set to true. This verifies that the .htaccess file is writable and works.

If it is not, then any options controlled by .htaccess, such as large file uploads, will not work. It also runs checks on the data/ directory, which verifies that it can't be accessed directly through the Web server.

'config_is_read_only' => false,

In certain environments it is desired to have a read-only configuration file.

When this switch is set to true ownCloud will not verify whether the configuration is writable. However, it will not be possible to configure all options via the Web interface. Furthermore, when updating ownCloud it is required to make the configuration file writable again for the update process.

5.8.9 Logging

'log_type' => 'owncloud',

By default the ownCloud logs are sent to the owncloud.log file in the default ownCloud data directory.

If syslogging is desired, set this parameter to syslog. Setting this parameter to errorlog will use the PHP error_log function for logging.

'logfile' => '/var/log/owncloud.log',

Log file path for the ownCloud logging type.

Defaults to [datadirectory]/owncloud.log

'loglevel' => 2,

Loglevel to start logging at. Valid values are: 0 = Debug, 1 = Info, 2 = Warning, 3 = Error, and 4 = Fatal. The default value is Warning.

```
'syslog_tag' => 'ownCloud',
```

If you maintain different instances and aggregate the logs, you may want to distinguish between them. syslog_tag can be set per instance with a unique id. Only available if log_type is set to syslog.

The default value is ownCloud.

```
'log.condition' => [
    'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
    'users' => ['sample-user'],
    'apps' => ['files'],
],
```

Log condition for log level increase based on conditions. Once one of these conditions is met, the required log level is set to debug. This allows to debug specific requests, users or apps

Supported conditions:

- shared_secret: if a request parameter with the name *log_secret* is set to this value the condition is met
- users: if the current request is done by one of the specified users, this condition is met
- apps: if the log message is invoked by one of the specified apps, this condition is met

Defaults to an empty array.

'logdateformat' => 'F d, Y H:i:s',

This uses PHP.date formatting; see http://php.net/manual/en/function.date.php

'logtimezone' => 'Europe/Berlin',

The default timezone for logfiles is UTC. You may change this; see http://php.net/manual/en/timezones.php

'log_query' => false,

Append all database queries and parameters to the log file. Use this only for debugging, as your logfile will become huge.

```
'cron_log' => true,
```

Log successful cron runs.

```
'cron.lockfile.location' => '',
```

Location of the lock file for cron executions can be specified here.

Default is within the tmp directory. The file is named in the following way: owncloud-server-\$INSTANCEID cron.lock where \$INSTANCEID is the string specified in the instanceid field. Because the cron lock file is accessed at regular intervals, it may prevent enabled disk drives from spinning down. A different location for this file can solve such issues.

'log_rotate_size' => false,

Enables log rotation and limits the total size of logfiles. The default is 0, or no rotation. Specify a size in bytes, for example 104857600 (100 megabytes = 100 * 1024 * 1024 bytes). A new logfile is created with a new name when the old logfile reaches your limit. If a rotated log file is already present, it will be overwritten.

5.8.10 Alternate Code Locations

Some of the ownCloud code may be stored in alternate locations.

'3rdpartyroot' => '',

ownCloud uses some 3rd party PHP components to provide certain functionality.

These components are shipped as part of the software package and reside in owncloud/3rdparty. Use this option to configure a different location. For example, if your location is /var/www/owncloud/foo/3rdparty, then the correct configuration is '3rdpartyroot' => '/var/www/owncloud/foo/',

```
'3rdpartyurl' => '',
```

If you have an alternate 3rdpartyroot, you must also configure the URL as seen by a Web browser.

This section is for configuring the download links for ownCloud clients, as seen in the first-run wizard and on Personal pages.

5.8.11 Apps

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

'appstoreenabled' => true,

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

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

The URL of the appstore to use.

'appstore.experimental.enabled' => false,

Whether to show experimental apps in the appstore interface

Experimental apps are not checked for security issues and are new or known to be unstable and under heavy development. Installing these can cause data loss or security breaches.

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

ownCloud supports previews of image files, the covers of MP3 files, and text files. These options control enabling and disabling previews, and thumbnail size.

'enable_previews' => true,

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

- Image files
- · Covers of MP3 files
- · Text documents

Valid values are true, to enable previews, or false, to disable previews

```
'preview_max_x' => 2048,
```

The maximum width, in pixels, of a preview. A value of null means there is no limit.

'preview_max_y' => 2048,

The maximum height, in pixels, of a preview. A value of null means there is no limit.

'preview_max_scale_factor' => 10,

If a lot of small pictures are stored on the ownCloud instance and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, pictures are upscaled to 10 times the original size. A value of 1 or null disables scaling.

'preview_max_filesize_image' => 50,

max file size for generating image previews with imagegd (default behaviour) If the image is bigger, it'll try other preview generators, but will most likely show the default mimetype icon

Value represents the maximum filesize in megabytes Default is 50 Set to -1 for no limit

'preview_libreoffice_path' => '/usr/bin/libreoffice',

custom path for LibreOffice/OpenOffice binary

Use this if LibreOffice/OpenOffice requires additional arguments.

```
'enabledPreviewProviders' => array(
    'OC\Preview\PNG',
    'OC\Preview\JPEG',
    'OC\Preview\GIF',
    'OC\Preview\BMP',
    'OC\Preview\XBitmap',
    'OC\Preview\MP3',
    'OC\Preview\TXT',
    'OC\Preview\MarkDown'
```

),

Only register providers that have been explicitly enabled

The following providers are enabled by default:

- OC\Preview\PNG
- OC\Preview\JPEG
- OC\Preview\GIF
- OC\Preview\BMP
- OC\Preview\XBitmap
- OC\Preview\MarkDown
- OC\Preview\MP3
- OC\Preview\TXT

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

- OC\Preview\Illustrator
- OC\Preview\Movie
- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007
- OC\Preview\MSOfficeDoc
- OC\Preview\OpenDocument
- OC\Preview\PDF
- OC\Preview\Photoshop
- OC\Preview\Postscript
- OC\Preview\StarOffice
- OC\Preview\SVG
- OC\Preview\TIFF
- OC\Preview\Font

Note: Troubleshooting steps for the MS Word previews are available at the *Configuring the Collaborative Documents App* section of the Administrators Manual.

The following providers are not available in Microsoft Windows:

- OC\Preview\Movie
- OC\Preview\MSOfficeDoc
- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007
- OC\Preview\OpenDocument
- OC\Preview\StarOffice

5.8.13 LDAP

Global settings used by LDAP User and Group Backend

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

defines the interval in minutes for the background job that checks user existence and marks them as ready to be cleaned up. The number is always minutes. Setting it to 0 disables the feature.

See command line (occ) methods ldap:show-remnants and user:delete

5.8.14 Comments

Global settings for the Comments infrastructure

'comments.managerFactory' => '\OC\Comments\ManagerFactory',

Replaces the default Comments Manager Factory. This can be utilized if an own or 3rdParty CommentsManager should be used that – for instance – uses the filesystem instead of the database to keep the comments.

'systemtags.managerFactory' => '\OC\SystemTag\ManagerFactory',

Replaces the default System Tags Manager Factory. This can be utilized if an own or 3rdParty SystemTagsManager should be used that – for instance – uses the filesystem instead of the database to keep the comments.

5.8.15 Maintenance

These options are for halting user activity when you are performing server maintenance.

```
'maintenance' => false,
```

Enable maintenance mode to disable ownCloud

If you want to prevent users from logging in to ownCloud before you start doing some maintenance work, you need to set the value of the maintenance parameter to true. Please keep in mind that users who are already logged-in are kicked out of ownCloud instantly.

'singleuser' => false,

When set to true, the ownCloud instance will be unavailable for all users who are not in the admin group.

5.8.16 SSL

Extra SSL options to be used for configuration.

```
'enable_certificate_management' => false,
```

Allow the configuration of system wide trusted certificates

5.8.17 Memory caching backend configuration

Available cache backends:

- \OC\Memcache\APC Alternative PHP Cache backend
- \OC\Memcache\APCu APC user backend
- \OC\Memcache\ArrayCache In-memory array-based backend (not recommended)

- $OC\Memcache \Memcached Memcached backend$
- \OC\Memcache\Redis Redis backend
- \OC\Memcache\XCache backend

Advice on choosing between the various backends:

- APCu should be easiest to install. Almost all distributions have packages. Use this for single user environment for all caches.
- Use Redis or Memcached for distributed environments. For the local cache (you can configure two) take APCu.

'memcache.local' => '\OC\Memcache\APCu',

Memory caching backend for locally stored data

• Used for host-specific data, e.g. file paths

'memcache.distributed' => '\OC\Memcache\Memcached',

Memory caching backend for distributed data

- Used for installation-specific data, e.g. database caching
- If unset, defaults to the value of memcache.local

Connection details for redis to use for memory caching.

For enhanced security it is recommended to configure Redis to require a password. See http://redis.io/topics/security for more information.

Server details for one or more memcached servers to use for memory caching.

'cache_path' => '',

Location of the cache folder, defaults to data/\$user/cache where \$user is the current user. When specified, the format will change to \$cache_path/\$user where \$cache_path is the configured cache directory and \$user is the user.

5.8.18 Using Object Store with ownCloud

```
'username' => 'facebook100000123456789',
                // in the trystack dashboard go to user -> settings -> API Password to
                // generate a password
                'password' => 'Secr3tPaSSWoRdt7',
                // must already exist in the objectstore, name can be different
                'container' => 'owncloud',
                // create the container if it does not exist. default is false
                'autocreate' => true,
                // required, dev-/trystack defaults to 'RegionOne'
                'region' => 'RegionOne',
                // The Identity / Keystone endpoint
                'url' => 'http://8.21.28.222:5000/v2.0',
                // required on dev-/trystack
                'tenantName' => 'facebook100000123456789',
                // dev-/trystack uses swift by default, the lib defaults to 'cloudFiles'
                // if omitted
                'serviceName' => 'swift',
                // The Interface / url Type, optional
                'urlType' => 'internal'
        ),
),
```

This example shows how to configure ownCloud to store all files in a swift object storage.

It is important to note that ownCloud in object store mode will expect exclusive access to the object store container because it only stores the binary data for each file. The metadata is currently kept in the local database for performance reasons.

WARNING: The current implementation is incompatible with any app that uses direct file IO and circumvents our virtual filesystem. That includes Encryption and Gallery. Gallery will store thumbnails directly in the filesystem and encryption will cause severe overhead because key files need to be fetched in addition to any requested file.

One way to test is applying for a trystack account at http://trystack.org/

5.8.19 Sharing

Global settings for Sharing

```
'sharing.managerFactory' => '\OC\Share20\ProviderFactory',
```

Replaces the default Share Provider Factory. This can be utilized if own or 3rdParty Share Providers be used that – for instance – uses the filesystem instead of the database to keep the share information.

5.8.20 All other configuration options

Additional driver options for the database connection, eg. to enable SSL encryption in MySQL.

'sqlite.journal_mode' => 'DELETE',

sqlite3 journal mode can be specified using this configuration parameter - can be 'WAL' or 'DELETE' see for more details https://www.sqlite.org/wal.html

```
'supportedDatabases' => array(
    'sqlite',
    'mysql',
    'pgsql',
    'oci',
),
```

Database types that are supported for installation.

Available:

- sqlite (SQLite3 Not in Enterprise Edition)
- mysql (MySQL)
- pgsql (PostgreSQL)
- oci (Oracle Enterprise Edition Only)

'tempdirectory' => '/tmp/owncloudtemp',

Override where ownCloud stores temporary files. Useful in situations where the system temporary directory is on a limited space ramdisk or is otherwise restricted, or if external storages which do not support streaming are in use.

The Web server user must have write access to this directory.

```
'hashingCost' => 10,
```

The hashing cost used by hashes generated by ownCloud Using a higher value requires more time and CPU power to calculate the hashes

'blacklisted_files' => array('.htaccess'),

Blacklist a specific file or files and disallow the upload of files with this name. .htaccess is blocked by default.

WARNING: USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

```
'share_folder' => '/',
```

Define a default folder for shared files and folders other than root.

```
'theme' => '',
```

If you are applying a theme to ownCloud, enter the name of the theme here.

The default location for themes is owncloud/themes/.

'cipher' => 'AES-256-CFB',

The default cipher for encrypting files. Currently AES-128-CFB and AES-256-CFB are supported.

```
'minimum.supported.desktop.version' => '1.7.0',
```

The minimum ownCloud desktop client version that will be allowed to sync with this server instance. All connections made from earlier clients will be denied by the server. Defaults to the minimum officially supported ownCloud version at the time of release of this server version.

When changing this, note that older unsupported versions of the ownCloud desktop client may not function as expected, and could lead to permanent data loss for clients or other unexpected results.

'quota_include_external_storage' => false,

EXPERIMENTAL: option whether to include external storage in quota calculation, defaults to false.

'filesystem_check_changes' => 0,

Specifies how often the local filesystem (the ownCloud data/ directory, and NFS mounts in data/) is checked for changes made outside ownCloud. This does not apply to external storages.

0 -> Never check the filesystem for outside changes, provides a performance increase when it's certain that no changes are made directly to the filesystem

1 -> Check each file or folder at most once per request, recommended for general use if outside changes might happen.

'part_file_in_storage' => true,

On default ownCloud will store the part files created during upload in the same storage as the upload target. Setting this to false will store the part files in the root of the users folder which might be required to work with certain external storage setups that have limited rename capabilities.

'asset-pipeline.enabled' => false,

All css and js files will be served by the Web server statically in one js file and one css file if this is set to true. This improves performance.

'assetdirectory' => '/var/www/owncloud',

The parent of the directory where css and js assets will be stored if pipelining is enabled; this defaults to the ownCloud directory. The assets will be stored in a subdirectory of this directory named 'assets'. The server *must* be configured to serve that directory as \$WEBROOT/assets.

You will only likely need to change this if the main ownCloud directory is not writeable by the Web server in your configuration.

'mount_file' => '/var/www/owncloud/data/mount.json',

Where mount.json file should be stored, defaults to data/mount.json in the ownCloud directory.

'filesystem_cache_readonly' => false,

When true, prevent ownCloud from changing the cache due to changes in the filesystem for all storage.

'secret' => '',

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

'trusted_proxies' => array('203.0.113.45', '198.51.100.128'),

List of trusted proxy servers

If you configure these also consider setting *forwarded_for_headers* which otherwise defaults to *HTTP_X_FORWARDED_FOR* (the *X-Forwarded-For* header).

'forwarded_for_headers' => array('HTTP_X_FORWARDED', 'HTTP_FORWARDED_FOR'),

Headers that should be trusted as client IP address in combination with *trusted_proxies*. If the HTTP header looks like 'X-Forwarded-For', then use 'HTTP_X_FORWARDED_FOR' here.

If set incorrectly, a client can spoof their IP address as visible to ownCloud, bypassing access controls and making logs useless!

Defaults to 'HTTP_X_FORWARED_FOR' if unset

```
'max_filesize_animated_gifs_public_sharing' => 10,
```

max file size for animating gifs on public-sharing-site.

If the gif is bigger, it'll show a static preview

Value represents the maximum filesize in megabytes. Default is 10. Set to -1 for no limit.

'filelocking.enabled' => true,

Enables transactional file locking.

This is enabled by default.

Prevents concurrent processes from accessing the same files at the same time. Can help prevent side effects that would be caused by concurrent operations. Mainly relevant for very large installations with many users working with shared files.

'memcache.locking' => '\\OC\\Memcache\\Redis',

Memory caching backend for file locking

Because most memcache backends can clean values without warning using redis is highly recommended to *avoid data loss*.

```
'debug' => false,
```

Set this ownCloud instance to debugging mode

Only enable this for local development and not in production environments This will disable the minifier and outputs some additional debug information

'copied_sample_config' => true,

This entry is just here to show a warning in case somebody copied the sample configuration. DO NOT ADD THIS SWITCH TO YOUR CONFIGURATION!

If you, brave person, have read until here be aware that you should not modify *ANY* settings in this file without reading the documentation.

5.8.21 App config options

Retention for activities of the activity app:

```
'activity_expire_days' => 365,
```

Every day a cron job is ran, which deletes all activities for all users which are older then the number of days that is set for activity_expire_days

'wnd.logging.enable' => true,

This enables debug logs for the windows_network_drive app.

5.9 Email Configuration

ownCloud is capable of sending password reset emails, notifying users of new file shares, changes in files, and activity notifications. Your users configure which notifications they want to receive on their Personal pages.

ownCloud does not contain a full email server, but rather connects to your existing mail server. You must have a functioning mail server for ownCloud to be able to send emails. You may have a mail server on the same machine as ownCloud, or it may be a remote server.

ownCloud 7 introduces a new feature, the graphical Email Configuration Wizard.

Email Server

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



With the new wizard, connecting ownCloud to your mail server is fast and easy. The wizard fills in the values in config/config.php, so you may use either or both as you prefer.

The ownCloud Email wizard supports three types of mail server connections: SMTP, PHP, and Sendmail. Use the SMTP configurator for a remote server, and PHP or Sendmail when your mail server is on the same machine as ownCloud.

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

5.9.1 Configuring an SMTP Server

You need the following information from your mailserver administrator to connect ownCloud to a remote SMTP server:

- Encryption type: None, SSL/TLS or STARTTLS
- The From address you want your outgoing ownCloud mails to use
- Whether authentication is required
- Authentication method: None, Login, Plain, or NT LAN Manager
- The server's IP address or fully-qualified domain name
- Login credentials, if required

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

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

| Email Server | | | | | | | | |
|--|----------------|---|--|--|--|--|--|--|
| This is used for sending out notifications. Saving | | | | | | | | |
| Send mode | smtp | Encryption TLS 🔹 | | | | | | |
| From address | owncloud | @ alrac.net | | | | | | |
| Authentication method | Login | Authentication required | | | | | | |
| Server address | None
Login | : Port | | | | | | |
| Credentials | NT LAN Manager | | | | | | | |
| Test email settings Send email | | | | | | | | |

ownCloud web services under your control

5.9.2 Configuring PHP and Sendmail

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

| Email Server | | | | | | | | |
|---|--|--|--|--|--|--|--|--|
| This is used for sending out notifications. Sav | This is used for sending out notifications. Saving | | | | | | | |
| Send mode | sendmail 🔹 | | | | | | | |
| From address | owncloud @ alrac.net | | | | | | | |
| Test email settings Send email | | | | | | | | |

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

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

5.9.3 Using Email Templates

Another useful new feature is editable email templates. Now you can edit ownCloud's email templates on your Admin page. These are your available templates:

- Sharing email (HTML) HTML version of emails notifying users of new file shares
- Sharing email (plain text fallback) Plain text email notifying users of new file shares
- Lost password mail Password reset email for users who lose their passwords.
- Activity notification mail Notification of activities that users have enabled in the Notifications section of their Personal pages.

In addition to providing the email templates, this feature enables you to apply any preconfigured themes to the email.

To modify an email template to users:

- 1. Access the Admin page.
- 2. Scroll to the Mail templates section.
- 3. Select a template from the drop-down menu.
- 4. Make any desired modifications to the template.

The templates are written in PHP and HTML, and are already loaded with the relevant variables such as username, share links, and filenames. You can, if you are careful, edit these even without knowing PHP or HTML; don't touch any of the code, but you can edit the text portions of the messages. For example, this the lost password mail template:

<?php

```
echo str_replace('{link}', $_['link'], $1->t('Use the following link to
reset your password: {link}'));
```

You could change the text portion of the template, Use the following link to reset your password: to say something else, such as Click the following link to reset your password. If you did not ask for a password reset, ignore this message.

Again, be very careful to change nothing but the message text, because the tiniest coding error will break the template.

Note: You can edit the templates directly in the template text box, or you can copy and paste them to a text editor for modification and then copy and paste them back to the template text box for use when you are done.

5.9.4 Setting Mail Server Parameters in config.php

If you prefer, you may set your mail server parameters in config/config.php. The following examples are for SMTP, PHP, Sendmail, and Qmail.

SMTP

If you want to send email using a local or remote SMTP server it is necessary to enter the name or IP address of the server, optionally followed by a colon separated port number, e.g. :425. If this value is not given the default port 25/tcp will be used unless you change that by modifying the **mail_smtpport** parameter. Multiple servers can be entered, separated by semicolons:

<?php

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

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

<?php

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

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

<?php

"mail_smtpsecure" => '',

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

SSL/TLS

A secure connection will be initiated using SSL/TLS via SMTPS on the default port 465/tcp:

<?php

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

STARTTLS

A secure connection will be initiated using STARTTLS via SMTP on the default port 25/tcp:

<?php

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

An alternative is the port 587/tcp (recommended):

<?php

```
"mail_smtphost" => "smtp.server.dom:587",
"mail_smtpsecure" => 'tls',
```

Authentication

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

<?php

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

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

<?php

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

PHP mail

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

<?php

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

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" => "",
```

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" => "",
```

5.9.5 Send a Test Email

To test your email configuration, save your email address in your personal settings and then use the **Send email** button in the *Email Server* section of the Admin settings page.

5.9.6 Using Self-Signed Certificates

When using self-signed certificates on the remote SMTP server the certificate must be imported into ownCloud. Please refer to *Importing System-wide and Personal SSL Certificates* for more information.

5.9.7 Troubleshooting

If you are unable to send email, try turning on debugging. Do this by enabling the mail_smtpdebug parameter in config/config.php.

<?php

"mail_smtpdebug" => true;

Note: Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines():** ... messages appear on the screen. This is expected behavior and can be ignored.

Question: Why is my web domain different from my mail domain?

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

<?php

"mail_domain" => "example.com",

This setting results in every email sent by ownCloud (for example, the password reset email) having the domain part of the sender address appear as follows:

no-reply@example.com

Question: How can I find out if an SMTP server is reachable?

Answer: Use the ping command to check the server availability:

ping smtp.server.dom

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

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

Answer: The best way to get mail server information is to ask your mail server admin. If you are the mail server admin, or need information in a hurry, you can use the netstat command. This example shows all active servers on your system, and the ports they are listening on. The SMTP server is listening on localhost port 25.

netstat -pant

```
Active Internet connections (servers and established)Proto Recv-Q Send-Q Local AddressForeign AddressStateID/Program nametcp00.0.0.0:6310.0.0.0:*LISTEN4418/cupsdtcp00127.0.0.1:250.0.0.0:*LISTEN2245/exim4tcp00127.0.0.1:33060.0.0.0:*LISTEN1524/mysqld
```

- 25/tcp is unencrypted smtp
- 110/tcp/udp is unencrypted pop3
- 143/tcp/udp is unencrypted imap4
- 465/tcp is encrypted smtps
- 993/tcp/udp is encrypted imaps
- 995/tcp/udp is encrypted pop3s

Question: How can I determine if the SMTP server supports SMTPS?

Answer: A good indication that the SMTP server supports SMTPS is that it is listening on port 465.

Question: How can I determine what authorization and encryption protocols the mail server supports?

Answer: SMTP servers usually announce the availability of STARTTLS immediately after a connection has been established. You can easily check this using the telnet command.

Note: You must enter the marked lines to obtain the information displayed.

250 HELP QUIT # <<< enter this command 221 smtp.domain.dom closing connection Connection closed by foreign host.

5.9.8 Enabling Debug Mode

If you are unable to send email, it might be useful to activate further debug messages by enabling the mail_smtpdebug parameter:

<?php

"mail_smtpdebug" => true,

Note: Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines():** ... messages appear on the screen. This is expected behavior and can be ignored.

5.10 Linking External Sites

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



Figure 5.2: Click to enlarge

This is useful for quick access to important Web pages such as the ownCloud manuals and informational pages for your company, and for presenting external pages inside your custom ownCloud branding, if you use your own custom themes.

The External sites app is included in all versions of ownCloud. Go to **Apps > Not Enabled** to enable it. Then go to your ownCloud Admin page to create your links, which are saved automatically. There is a dropdown menu to select an icon, but there is only one default icon so you don't have to select one. Hover your cursor to the right of your links to make the trashcan icon appear when you want to remove them.

The links appear in the ownCloud dropdown menu on the top left after refreshing your page, and have globe icons.

| External Sites | | | | | | | |
|---|---|--|--|--|--|--|--|
| Please note that some
Furthermore please no
We highly recommend | browsers will block display
ote that many sites these da
I to test the configured sites | ing of sites via HTTP if you are running HTTPS.
ays disallow iframing due to security reasons.
s below properly. | | | | | |
| ownCloud docs https://doc.owncloud external.png | | | | | | | |
| Add External sites | s saved. | • | | | | | |





Your links may or may not work correctly due to the various ways that Web browsers and Web sites handle HTTP and HTTPS URLs, and because the External Sites app embeds external links in IFrames. Modern Web browsers try very hard to protect Web surfers from dangerous links, and safety apps like Privacy Badger and ad-blockers may block embedded pages. It is strongly recommended to enforce HTTPS on your ownCloud server; do not weaken this, or any of your security tools, just to make embedded Web pages work. After all, you can freely access them outside of ownCloud.

Most Web sites that offer login functionalities use the X-Frame-Options or Content-Security-Policy HTTP header which instructs browsers to not allow their pages to be embedded for security reasons (e.g. "Clickjack-ing"). You can usually verify the reason why embedding the website is not possible by using your browser's console tool. For example, this page has an invalid SSL certificate.

| ownCloud - ownCloud × | Lukas |
|---|--------------|
| ← → C 🗋 localhost/master/index.php/apps/external/1 👷 🕄 🍪 🎈 🧏 🥞 🔕 | ⓐ ≡ |
| م ad | min 🔻 |
| | |
| | |
| 🔍 🛛 Elements Network Sources Timeline Profiles Resources Audits Console » 🛛 😒 2 🔀 | • 💷 🗸 × |
| S T <top frame=""> ▼ □ Preserve log</top> | |
| Set http://localhost/master/core/vendor/jquery/dist/jquery-migrate.min.map 404 (Not Found) | 1:1 |
| <pre> GET <u>https://invalid.owncloud.org/</u> net::ERR_INSECURE_RESPONSE > </pre> | <u>1:184</u> |

Console Search Emulation Rendering

On this page, X-Frame-Options prevents the embedding.

There isn't much you can do about these issues, but if you're curious you can see what is happening.

5.11 Custom Client Download Repositories

You may configure the URLs to your own download repositories for your ownCloud desktop clients and mobile apps in config/config.php. This example shows the default download locations:



| Q | . 🛛 | Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console » | 🛛 2 🗲 | \$ | $\Box \Box_{a} \times$ |
|---|--------------|--|------------------------|-----------|------------|------------|-------------|-----------|-------------------|------------|-------|------------------------|
| 0 |) F | ′∕ <top fra<="" th=""><th>me></th><th></th><th>▼ □ P</th><th>reserve lo</th><th>g</th><th></th><th></th><th></th><th></th><th></th></top> | me> | | ▼ □ P | reserve lo | g | | | | | |
| 0 | GET | http://loc | alhost/ma | ster/core | e/vendor/j | query/di | lst/jquery- | migrate.m | <u>in.map</u> 404 | (Not Found |) | 1:1 |
| 8 | Refu
Opti | sed to dis
ons' to 'S | play 'htt
AMEORIGIN | ps://owno | cloud.com/ | 'in a f | rame becau | se it set | 'X-Frame- | abo | ut:bl | ank:1 |
| ~ | | | | | | | | | | | | |

Console Search Emulation Rendering

<?php

```
"customclient_desktop" => "https://owncloud.org/sync-clients/",
"customclient_android" => "https://play.google.com/store/apps/details?id=com.owncloud.android",
"customclient_ios" => "https://itunes.apple.com/us/app/owncloud/id543672169?mt=8",
```

Simply replace the URLs with the links to your own preferred download repos.

You may test alternate URLs without editing config/config.php by setting a test URL as an environment variable:

export OCC_UPDATE_URL=https://test.example.com

When you're finished testing you can disable the environment variable:

unset OCC_UPDATE_URL

5.12 Knowledge Base Configuration

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

5.12.1 Parameters

If you want to disable the ownCloud help menu item you can use the **knowledgebaseenabled** parameter inside the config/config.php.

<?php

```
"knowledgebaseenabled" => true,
```

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

5.13 Language Configuration

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

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

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

5.13.1 Parameters

<?php

"default_language" => "en",

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

5.14 Logging Configuration

Use your ownCloud log to review system status, or to help debug problems. You may adjust logging levels, and choose between using the ownCloud log or your syslog.

5.14.1 Parameters

Logging levels range from **DEBUG**, which logs all activity, to **FATAL**, which logs only fatal errors.

- 0: DEBUG: All activity; the most detailed logging.
- 1: INFO: Activity such as user logins and file activities, plus warnings, errors, and fatal errors.
- 2: WARN: Operations succeed, but with warnings of potential problems, plus errors and fatal errors.
- 3: ERROR: An operation fails, but other services and operations continue, plus fatal errors.
- 4: FATAL: The server stops.

By default the log level is set to 2 (WARN). Use **DEBUG** when you have a problem to diagnose, and then reset your log level to a less-verbose level as **DEBUG** outputs a lot of information, and can affect your server performance.

Logging level parameters are set in the config/config.php file, or on the Admin page of your ownCloud Web GUI.

ownCloud

All log information will be written to a separate log file which can be viewed using the log viewer on your Admin page. By default, a log file named **owncloud.log** will be created in the directory which has been configured by the **datadirectory** parameter in config.php.

The desired date format can optionally be defined using the **logdateformat** parameter in config/config.php. By default the PHP date function parameter "c" is used, and therefore the date/time is written in the format "2013-01-10T15:20:25+02:00". By using the date format in the example below, the date/time format will be written in the format "January 10, 2013 15:20:25".

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

syslog

All log information will be sent to your default syslog daemon.

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

Conditional Logging Level Increase

You can configure the logging level to automatically increase to debug when one of three conditions are met:

shared_secret: If a request parameter with the name log_secret is set to this value the condition is met.

users: If the current request is done by one of the specified users, this condition is met.

apps: If the log message is invoked by one of the specified apps, this condition is met.

The following example demonstrates what all three conditions look like:

```
'log.condition' => [
    'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
    'users' => ['sample-user'],
    'apps' => ['files'],
],
```

5.15 Hardening and Security Guidance

ownCloud aims to ship with secure defaults that do not need to get modified by administrators. However, in some cases some additional security hardening can be applied in scenarios were the administrator has complete control over the ownCloud instance. This page assumes that you run ownCloud Server on Apache2 in a Linux environment.

Note: ownCloud will warn you in the administration interface if some critical security-relevant options are missing. However, it is still up to the server administrator to review and maintain system security.

5.15.1 Limit on Password Length

ownCloud uses the bcrypt algorithm, and thus for security and performance reasons, e.g. Denial of Service as CPU demand increases exponentially, it only verifies the first 72 characters of passwords. This applies to all passwords that you use in ownCloud: user passwords, passwords on link shares, and passwords on external shares.

5.15.2 Operating system

Give PHP read access to /dev/urandom

ownCloud uses a RFC 4086 ("Randomness Requirements for Security") compliant mixer to generate cryptographically secure pseudo-random numbers. This means that when generating a random number ownCloud will request multiple random numbers from different sources and derive from these the final random number.

The random number generation also tries to request random numbers from /dev/urandom, thus it is highly recommended to configure your setup in such a way that PHP is able to read random data from it.

Note: When having an open_basedir configured within your php.ini file, make sure to include /dev/urandom.

Enable hardening modules such as SELinux

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

5.15.3 Deployment

Place data directory outside of the web root

It is highly recommended to place your data directory outside of the Web root (i.e. outside of /var/www). It is easiest to do this on a new installation.

Disable preview image generation

ownCloud is able to generate preview images of common filetypes such as images or text files. By default the preview generation for some file types that we consider secure enough for deployment is enabled by default. However, administrators should be aware that these previews are generated using PHP libraries written in C which might be vulnerable to attack vectors.

For high security deployments we recommend disabling the preview generation by setting the enable_previews switch to false in config.php. As an administrator you are also able to manage which preview providers are enabled by modifying the enabledPreviewProviders option switch.

5.15.4 Use HTTPS

Using ownCloud without using an encrypted HTTPS connection opens up your server to a man-in-the-middle (MITM) attack, and risks the interception of user data and passwords. It is a best practice, and highly recommended, to always use HTTPS on production servers, and to never allow unencrypted HTTP.

How to setup HTTPS on your Web server depends on your setup; please consult the documentation for your HTTP server. The following examples are for Apache.

Redirect all unencrypted traffic to HTTPS

To redirect all HTTP traffic to HTTPS administrators are encouraged to issue a permanent redirect using the 301 status code. When using Apache this can be achieved by adding a setting such as the following in the Apache VirtualHosts configuration containing the <VirtualHost *:80> entry:

Redirect permanent / https://example.com/

Enable HTTP Strict Transport Security

While redirecting all traffic to HTTPS is good, it may not completely prevent man-in-the-middle attacks. Thus administrators are encouraged to set the HTTP Strict Transport Security header, which instructs browsers to not allow any connection to the ownCloud instance using HTTP, and it attempts to prevent site visitors from bypassing invalid certificate warnings.

This can be achieved by setting the following settings within the Apache VirtualHost file containing the <VirtualHost *:443> entry:

```
<IfModule mod_headers.c>
Header always set Strict-Transport-Security "max-age=15768000; includeSubDomains"
</IfModule>
```

If you don't have access to your Apache configuration it is also possible to add this to the main .htaccess file shipped with ownCloud. Make sure you're adding it below the line:

DO NOT CHANGE ANYTHING ABOVE THIS LINE

This example configuration will make all subdomains only accessible via HTTPS. If you have subdomains not accessible via HTTPS, remove includeSubDomains.

Note: This requires the mod_headers extension in Apache.

When using nginx as a Web server an example is already included in the nginx Example Configurations:

#add_header Strict-Transport-Security "max-age=15768000; includeSubDomains";

You need to remove the # and reload nginx to enable this change.

Proper SSL configuration

Default SSL configurations by Web servers are often not state-of-the-art, and require fine-tuning for an optimal performance and security experience. The available SSL ciphers and options depend completely on your environment and thus giving a generic recommendation is not really possible.

We recommend using the Mozilla SSL Configuration Generator to generate a suitable configuration suited for your environment, and the free Qualys SSL Labs Tests gives good guidance on whether your SSL server is correctly configured.

Also ensure that HTTP compression is disabled to mitigate the BREACH attack.

5.15.5 Use a dedicated domain for ownCloud

Administrators are encouraged to install ownCloud on a dedicated domain such as cloud.domain.tld instead of domain.tld to gain all the benefits offered by the Same-Origin-Policy.

5.15.6 Ensure that your ownCloud instance is installed in a DMZ

As ownCloud supports features such as Federated File Sharing we do not consider Server Side Request Forgery (SSRF) part of our threat model. In fact, given all our external storage adapters this can be considered a feature and not a vulnerability.

This means that a user on your ownCloud instance could probe whether other hosts are accessible from the ownCloud network. If you do not want this you need to ensure that your ownCloud is properly installed in a segregated network and proper firewall rules are in place.

5.15.7 Serve security related Headers by the Web server

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

- X-Content-Type-Options: nosniff
 - Instructs some browsers to not sniff the mimetype of files. This is used for example to prevent browsers from interpreting text files as JavaScript.
- X-XSS-Protection: 1; mode=block
 - Instructs browsers to enable their browser side Cross-Site-Scripting filter.
- X-Robots-Tag: none
 - Instructs search machines to not index these pages.

```
• X-Frame-Options: SAMEORIGIN
```

- Prevents embedding of the ownCloud instance within an iframe from other domains to prevent Clickjacking and other similar attacks.

These headers are hard-coded into the ownCloud server, and need no intervention by the server administrator.

For optimal security, administrators are encouraged to serve these basic HTTP headers by the Web server to enforce them on response. To do this Apache has to be configured to use the .htaccess file and the following Apache modules need to be enabled:

- mod_headers
- mod_env

Administrators can verify whether this security change is active by accessing a static resource served by the Web server and verify that the above mentioned security headers are shipped.

5.16 Reverse Proxy Configuration

ownCloud can be run through a reverse proxy, which can cache static assets such as images, CSS or JS files, move the load of handling HTTPS to a different server or load balance between multiple servers.

5.16.1 Defining Trusted Proxies

For security, you must explicitly define the proxy servers that ownCloud is to trust. Connections from trusted proxies will be specially treated to get the real client information, for use in access control and logging. Parameters are configured in config/config.php

Set the **trusted_proxies** parameter as an array of IP address to define the servers ownCloud should trust as proxies. This parameter provides protection against client spoofing, and you should secure those servers as you would your ownCloud server.

A reverse proxy can define HTTP headers with the original client IP address, and ownCloud can use those headers to retrieve that IP address. ownCloud uses the de-facto standard header 'X-Forwarded-For' by default, but this can be configured with the **forwarded_for_headers** parameter. This parameter is an array of PHP lookup strings, for example 'X-Forwarded-For' becomes 'HTTP_X_FORWARDED_FOR'. Incorrectly setting this parameter may allow clients to spoof their IP address as visible to ownCloud, even when going through the trusted proxy! The correct value for this parameter is dependent on your proxy software.

5.16.2 Overwrite Parameters

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

If ownCloud fails to automatically detect the hostname, protocol or webroot you can use the **overwrite** parameters inside the config/config.php. The **overwritehost** parameter is used to set the hostname of the proxy. You can also specify a port. The **overwriteprotocol** parameter is used to set the protocol of the proxy. You can choose between the two options **http** and **https**. The **overwritewebroot** parameter is used to set the absolute web path of the proxy to the ownCloud folder. When you want to keep the automatic detection of one of the three parameters you can leave the value empty or don't set it. The **overwritecondaddr** parameter is used to overwrite the values dependent on the remote address. The value must be a **regular expression** of the IP addresses of the proxy. This is useful when you use a reverse SSL proxy only for https access and you want to use the automatic detection for http access.

5.16.3 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
$CONFIG = array (
    "trusted_proxies" => ['10.0.0.1'],
    "overwritehost" => "ssl-proxy.tld",
    "overwriteprotocol" => "https",
    "overwritewebroot" => "/domain.tld/owncloud",
    "overwritecondaddr" => "^10\.0\.0\.1$",
);
```

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

5.17 Using Third Party PHP Components

ownCloud uses some third party PHP components to provide some of its functionality. These components are part of the software package and are contained in the **/3rdparty** folder.

5.17.1 Managing Third Party Parameters

When using third party components, keep the following parameters in mind:

- **3rdpartyroot** Specifies the location of the 3rd-party folder. To change the default location of this folder, you can use this parameter to define the absolute file system path to the folder location.
- 3rdpartyurl Specifies the http web path to the 3rdpartyroot folder, starting at the ownCloud web root.

An example of what these parameters might look like is as follows:

<?php

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

5.18 JavaScript and CSS Asset Management

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

ownCloud can automatically collect all JavaScript and CSS files, aggregate and compress them to then save the result in a folder called 'assets' which can be found in the folder where ownCloud has been installed.

If your Web server has write access to your ownCloud installation, then the 'assets' folder will be automatically created for you, otherwise, you need to create it yourself before enabling that option and you must give write access to your Web server user.

Assets found in that folder will from now on be served as static files by your Web server and will be automatically refreshed whenever ownCloud or one of its apps is updated. It's important to note that apps installed via git might not always update their version number with every commit and this could lead to an out-of-sync asset folder. It is not recommended to enable asset-pipelining when using apps pulled via git.

5.18.1 Parameters

```
<?php
$CONFIG = array (
    ...
    'asset-pipeline.enabled' => true,
    ...
);
```

You can set this parameter in the config/config.php

5.19 Automatic Configuration Setup

If you need to install ownCloud on multiple servers, you normally do not want to set up each instance separately as described in *Database Configuration*. For this reason, ownCloud provides an automatic configuration feature.

To take advantage of this feature, you must create a configuration file, called .../owncloud/config/autoconfig.php, and set the file parameters as required. You can specify any number of parameters in this file. Any unspecified parameters appear on the "Finish setup" screen when you first launch ownCloud.

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

5.19.1 Parameters

When configuring parameters, you must understand that two parameters are named differently in this configuration file when compared to the standard config.php file.

| autoconfig.php | config.php |
|----------------|---------------|
| directory | datadirectory |
| dbpass | dbpassword |

5.19.2 Automatic Configurations Examples

The following sections provide sample automatic configuration examples and what information is requested at the end of the configuration.

Data Directory

Using the following parameter settings, the "Finish setup" screen requests database and admin credentials settings.

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

SQLite Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

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

MySQL Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

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

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

PostgreSQL Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

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

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

All Parameters

Using the following parameter settings, because all parameters are already configured in the file, the ownCloud installation skips the "Finish setup" screen.

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

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

5.20 ownCloud Server Tuning

5.20.1 Using cron to perform background jobs

See Background Jobs for a description and the benefits.

5.20.2 Enable JavaScript and CSS Asset Management

See JavaScript and CSS Asset Management for a description and the benefits.

5.20.3 Caching

Caching improves performance by storing data, code, and other objects in memory. Memory cache configuration for the ownCloud server is no longer automatic in ownCloud 8.1 and up, but must be installed and configured. See *Configuring Memory Caching*.

5.20.4 Using MariaDB/MySQL instead of SQLite

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

See the section *Database Configuration* for how to configure ownCloud for MySQL or MariaDB. If your installation is already running on SQLite then it is possible to convert to MySQL or MariaDB using the steps provided in *Converting Database Type*.

5.20.5 Using Redis-based Transactional File Locking

File locking is enabled by default, using the database locking backend. This places a significant load on your database. See the section *Transactional File Locking* for how to configure ownCloud to use Redis-based Transactional File Locking.

5.20.6 SSL / Encryption App

SSL (HTTPS) and file encryption/decryption can be offloaded to a processor's AES-NI extension. This can both speed up these operations while lowering processing overhead. This requires a processor with the AES-NI instruction set.

Here are some examples how to check if your CPU / environment supports the AES-NI extension:

- For each CPU core present: grep flags /proc/cpuinfo or as a summary for all cores: grep -m 1 ^flags /proc/cpuinfo If the result contains any aes, the extension is present.
- Search eg. on the Intel web if the processor used supports the extension Intel Processor Feature Filter You may set a filter by "AES New Instructions" to get a reduced result set.
- For versions of openssl >= 1.0.1, AES-NI does not work via an engine and will not show up in the openssl engine command. It is active by default on the supported hardware. You can check the openssl version via openssl version -a
- If your processor supports AES-NI but it does not show up eg via grep or coreinfo, it is maybe disabled in the BIOS.
- If your environment runs virtualized, check the virtualization vendor for support.

5.21 Enable index.php-less URLs

Since ownCloud 9.0.3 you need to explicitly configure and enable index.php-less URLs (e.g. https://example.com/apps/files/ instead of https://example.com/index.php/apps/files/). The following documentation provides the needed steps to configure this for the Apache Web server. These steps are not necessary when using nginx as a web server because it is already enabled in the *nginx Example Configurations*.

5.21.1 Prerequisites

Before being able to use index.php-less URLs you need to enable the mod_rewrite and mod_env Apache modules. Furthermore a configured AllowOverride All directive within the vhost of your Web server is needed. Please have a look at the Apache manual for how to enable and configure these.

Furthermore these instructions are only working when using Apache together with the mod_php Apache module for PHP. Other modules like php-fpm or mod_fastcgi are unsupported.

Finally the user running your Web server (e.g. www-data) needs to be able to write into the .htaccess file shipped within the ownCloud root directory (e.g. /var/www/owncloud/.htaccess). If you have applied *Setting Strong Directory Permissions* the user might be unable to write into this file and the needed update will fail. You need to revert this strong permissions temporarily by following the steps described in *Setting Permissions for Updating*.

5.21.2 Configuration steps

The first step is to configure the overwrite.cli.url and htaccess.RewriteBase config.php options (See *Config.php Parameters*). If you're accessing your ownCloud instance via https://example.com/ the following two options need to be added / configured:

```
'overwrite.cli.url' => 'https://example.com',
'htaccess.RewriteBase' => '/',
```

If the instance is accessed via https://example.com/owncloud the following configuration is needed:

```
'overwrite.cli.url' => 'https://example.com/owncloud',
'htaccess.RewriteBase' => '/owncloud',
```

As a second step ownCloud needs to enable index.php-less URLs. This is done:

- during the next update of your ownCloud instance
- by manually running the occ command occ maintenance:update:htaccess (See Using the occ Command)

Afterwards your instance should have index.php-less URLs enabled.

5.21.3 Troubleshooting

If accessing your ownCloud installation fails after following these instructions and you see messages like this in your ownCloud log:

```
The requested uri(\\/login) cannot be processed by the script '\\/owncloud\\/index.php'
```

make sure that you have configured the two config.php options listed above correctly.

CHAPTER

SIX

USER MANAGEMENT

6.1 User Management

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

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

The default view displays basic information about your users.

| Usern | ame | Password | Groups | • Create | | Search Users |
|-------|----------|-----------|----------|------------------|-----------------|--------------|
| | Username | Full Name | Password | Groups | Group Admin for | Quota |
| Α | admin | admin | ••••• | admin 🔻 | no group 🔹 | Default - |
| 2 | layla | layla | ••••• | users, artists 🔻 | artists 🔻 | 10 GB 🔹 |
| | molly | molly | ••••• | users v | no group 🔻 | Default - |
| A | ritasue | ritasue | ••••• | artists v | users v | 10 GB 🔹 |
| 1 | stashcat | stashcat | ••••• | users, admin 🔻 | no group 🔹 | 5 GB 🔹 |

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

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

| + Add Group | |
|-------------|---|
| Everyone | 5 |
| Admins | 2 |
| users | 3 |
| artists | 2 |



User accounts have the following properties:

Login Name (Username) The unique ID of an ownCloud user, and it cannot be changed.

- *Full Name* The user's display name that appears on file shares, the ownCloud Web interface, and emails. Admins and users may change the Full Name anytime. If the Full Name is not set it defaults to the login name.
- *Password* The admin sets the new user's first password. Both the user and the admin can change the user's password at anytime.
- *Groups* You may create groups, and assign group memberships to users. By default new users are not assigned to any groups.
- *Group Admin* Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.
- *Quota* The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. You have the the option to include external storage in user quotas.

6.1.1 Creating a New User

To create a user account:

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

| ÷., | | | | | | |
|-----|-------|----------|-----------|-------------|---|--------|
| | terry | ••• | •• | users | • | Create |
| | | Username | Full Name | 📃 admin | | rord |
| | Α | admin | admin | artists | | •••• |
| | 2 | layla | layla | 🖌 users | | •••• |
| | C. | molly | molly | + add group | | •••• |
| | A | ritasue | ritasue | (| | |

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

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

6.1.2 Reset a User's Password

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

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

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

6.1.3 Renaming a User

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

To set or change a user's display name:

- Hover your cursor over the user's Full Name field
- Click on the Pencil icon
- Enter the user's new display name

6.1.4 Granting Administrator Privileges to a User

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

| T | terry | terry | •••••• | redgroup 🔻 | Group Admin 🔻 |
|---|-------|-------|--------|------------|---------------|
| | | | | | 3group |
| | | | | | all-users |
| | | | | | group4 |
| | | | | | redgroup |

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

6.1.5 Managing Groups

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

6.1.6 Setting Storage Quotas

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

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

'quota_include_external_storage' => false,

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

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

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

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

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

6.1.7 Deleting users

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

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

6.2 Resetting a Lost Admin Password

The normal ways to recover a lost password are:

- 1. Click the password reset link on the login screen; this appears after a failed login attempt. This works only if you have entered your email address on your Personal page in the ownCloud Web interface, so that the ownCloud server can email a reset link to you.
- 2. Ask another ownCloud server admin to reset it for you.

If neither of these is an option, then you have a third option, and that is using the occ command. occ is in the owncloud directory, for example /var/www/owncloud/occ. occ has a command for resetting all user passwords, user:resetpassword. It is best to run occ as the HTTP user, as in this example on Ubuntu Linux:

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

If your ownCloud username is not admin, then substitute your ownCloud username.

You can find your HTTP user in your HTTP configuration file. These are the default Apache HTTP user: group on Linux distros:

- Centos, Red Hat, Fedora: apache:apache
- Debian, Ubuntu, Linux Mint: www-data:www-data
- openSUSE: wwwrun:www

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

6.3 Resetting a User Password

The ownCloud login screen displays a **Wrong password. Reset it?** message after a user enters an incorrect password, and then ownCloud automatically resets their password. However, if you are using a read-only authentication backend such as LDAP or Active Directory, this will not work. In this case you may specify a custom URL in your config.php file to direct your user to a server than can handle an automatic reset:

'lost_password_link' => 'https://example.org/link/to/password/reset',

6.4 User Authentication with IMAP, SMB, and FTP

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

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

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

6.4.1 IMAP

Provides authentication against IMAP servers

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

6.4.2 SMB

Provides authentication against Samba servers

- Class: OC_User_SMB
- Arguments: the samba server to authenticate against
- Dependency: PHP smbclient module or smbclient (see SMB/CIFS)
- Example:

```
<?php
```

6.4.3 FTP

Provides authentication against FTP servers

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

```
<?php
```

6.5 User Authentication with LDAP

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

Note: The PHP LDAP module is required; this is supplied by php5-ldap on Debian/Ubuntu, and php-ldap on CentOS/Red Hat/Fedora. PHP 5.4+ is required in ownCloud 8.1.

The LDAP application supports:

- LDAP group support
- File sharing with ownCloud users and groups
- · Access via WebDAV and ownCloud Desktop Client
- Versioning, external Storage and all other ownCloud features
- Seamless connectivity to Active Directory, with no extra configuration required
- Support for primary groups in Active Directory
- Auto-detection of LDAP attributes such as base DN, email, and the LDAP server port number
- Only read access to your LDAP (edit or delete of users on your LDAP is not supported)

Warning: The LDAP app is not compatible with the User backend using remote HTTP servers app. You cannot use both of them at the same time.

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

6.5.1 Configuration

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

The LDAP configuration panel has four tabs. A correctly completed first tab ("Server") is mandatory to access the other tabs. A green indicator lights when the configuration is correct. Hover your cursor over the fields to see some pop-up tooltips.

Server Tab

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

- Server configuration: Configure one or more LDAP servers. Click the Delete Configuration button to remove the active configuration.
- **Host:** The host name or IP address of the LDAP server. It can also be a **ldaps://** URI. If you enter the port number, it speeds up server detection.

LDAP

| Γ | Server Users Login Attributes Groups | Advanced | Expert |
|---|---|-------------|--------|
| | 1. Server 🔹 🕂 😰 | | |
| | Host Port Detect Port | | |
| | User DN | | |
| | Password | | |
| | One Base DN per line Detect Base DN T | est Base DN | |
| | Manually enter LDAP filters (recommended for large directories) | | |
| | | | |
| | Configuration incomplete Continue i Help | | |

Examples:

- directory.my-company.com
- ldaps://directory.my-company.com
- *directory.my-company.com*:9876
- **Port:** The port on which to connect to the LDAP server. The field is disabled in the beginning of a new configuration. If the LDAP server is running on a standard port, the port will be detected automatically. If you are using a non-standard port, ownCloud will attempt to detect it. If this fails you must enter the port number manually.

Example:

• 389

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

Example:

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

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

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

Example:

• *dc=my-company,dc=com*

User Filter

Use this to control which LDAP users are listed as ownCloud users on your ownCloud server. In order to control which LDAP users can login to your ownCloud server use the Login filter. Those LDAP users who have access but

are not listed as users (if there are any) will be hidden users. You may bypass the form fields and enter a raw LDAP filter if you prefer.

| Server | Users | Login Attributes | Groups | | | | Advanced | Ex |
|------------|--------------|---|--|---|----|---|----------|----|
| Limit own | | | | | | | | |
| Only thes | e object cla | sses: Select object | classes | 4 | ; | | | |
| | | The most com
are organizatio
inetOrgPerson
object class to
directory admi | mon object class
onalPerson, pers
. If you are not s
select, please c
n. | ses for users
on, user, and
sure which
onsult your | | | | |
| Only fro | m these gr | oups: Select groups | | 4 | , | | | |
| <u>1 E</u> | Edit LDAP (| Query | | | | | | |
| Edit LDA | P Query | | | | | | | |
| | | | | | | 2 | | |
| | | | | | | | | |
| Verify s | ettings and | count users | | | | | | |
| | Co | onfiguration incomplete | Back | continue i He | lp | | | |

- **only those object classes:** ownCloud will determine the object classes that are typically available for user objects in your LDAP. ownCloud will automatically select the object class that returns the highest amount of users. You may select multiple object classes.
- only from those groups: If your LDAP server supports the member-of-overlay in LDAP filters, you can define that only users from one or more certain groups are allowed to appear in user listings in ownCloud. By default, no value will be selected.

You may select multiple groups.

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

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

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

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

Login Filter

The settings in the Login Filter tab determine which LDAP users can log in to your ownCloud system and which attribute or attributes the provided login name is matched against (e.g. LDAP/AD username, email address). You may

select multiple user details. (You may bypass the form fields and enter a raw LDAP filter if you prefer.) You may override your User Filter settings on the User Filter tab by using a raw LDAP filter.

| Server | Users | Login Attributes | Groups | | | Advanced | Expert | | |
|------------|---|-------------------------|--------|-------------------|---|----------|--------|--|--|
| When log | When logging in, ownCloud will find the user based on the following attributes: | | | | | | | | |
| LDAP | LDAP / AD Usemame: 😥 | | | | | | | | |
| L | DAP / AD B | Email | | | | | | | |
| | Add | Iress: | | | | | | | |
| | Other Attrib | utes: Select attribut | ies | \$ | | | | | |
| <u>1 E</u> | dit LDAP C | Query | | | | | | | |
| Edit LDA | P Query | | | | | | | | |
| | | | | | 4 | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Test Log | inname | Verify settin | igs | | | | | | |
| | Co | onfiguration incomplete | Back | Continue i Help | | | | | |

- **LDAP Username:** If this value is checked, the login value will be compared to the username in the LDAP directory. The corresponding attribute, usually *uid* or *samaccountname* will be detected automatically by ownCloud.
- **LDAP Email Address:** If this value is checked, the login value will be compared to an email address in the LDAP directory; specifically, the *mailPrimaryAddress* and *mail* attributes.
- **Other Attributes:** This multi-select box allows you to select other attributes for the comparison. The list is generated automatically from the user object attributes in your LDAP server.

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

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

Examples:

• only username:

```
(&(objectClass=inetOrgPerson)(memberOf=cn=owncloudusers,ou=groups,dc=example,dc=com)(uid=%uid)
```

• username or email address:

```
((&(objectClass=inetOrgPerson) (memberOf=cn=owncloudusers,ou=groups,
dc=example,dc=com) (|(uid=%uid)(mail=%uid)))
```

Group Filter

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

| Server | Users | Login Attributes | Groups | | | Advanced | Expert | | |
|------------|--|-------------------------|---------|---------------|--|----------|--------|--|--|
| Groups m | Groups meeting these criteria are available in ownCloud: | | | | | | | | |
| Only thes | e object cla | sses: Select object | classes | \$ | | | | | |
| Only fro | m these gr | oups: Select groups | | \$ | | | | | |
| <u>1 E</u> | Edit LDAP (| Query | | | | | | | |
| Edit LDA | P Query | | | | | | | | |
| | | | | .1 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| Verify s | ettings and | I count groups | | | | | | | |
| | Co | onfiguration incomplete | Back | <i>i</i> Help | | | | | |

- **only those object classes:** ownCloud will determine the object classes that are typically available for group objects in your LDAP server. ownCloud will only list object classes that return at least one group object. You can select multiple object classes. A typical object class is "group", or "posixGroup".
- **only from those groups:** ownCloud will generate a list of available groups found in your LDAP server. and then you select the group or groups that get access to your ownCloud server.

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

Example:

- objectClass=group
- objectClass=posixGroup
- **y groups found:** This tells you approximately how many groups will be available in ownCloud. The number updates automatically after any change.

6.5.2 Advanced Settings

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

The Advanced Settings are structured into three parts:

- Connection Settings
- · Directory Settings
- Special Attributes

Connection Settings

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

LDAP

| Server | Users | Login Attributes | Groups | Ad | lvanced | Expert | | | |
|--------|--|---------------------|--------|----|---------|--------|--|--|--|
| - Con | Connection Settings | | | | | | | | |
| C | onfiguration
Active | | | | | | | | |
| (R | Backup
eplica) Host | | | | | | | | |
| (R | Backup
Replica) Port | | | | | | | | |
| D | isable Main
Server | | | | | | | | |
| I | Furn off SSL
certificate
validation. | | | | | | | | |
| т | Cache
ime-To-Live | 600 | | | | | | | |
| → Dire | ctory Setti | ngs | | | | | | | |
| → Spe | cial Attribu | tes | | | | | | | |
| Test C | onfiguratio | n ⁱ Help | | | | | | | |

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

Example:

- *directory2.my-company.com*
- **Backup (Replica) Port:** The connection port of the backup LDAP server. If no port is given, but only a host, then the main port (as specified above) will be used.

Example:

• 389

- **Disable Main Server:** You can manually override the main server and make ownCloud only connect to the backup server. This is useful for planned downtimes.
- Turn off SSL certificate validation: Turns off SSL certificate checking. Use it for testing only!
- **Cache Time-To-Live:** A cache is introduced to avoid unnecessary LDAP traffic, for example caching usernames so they don't have to be looked up for every page, and speeding up loading of the Users page. Saving the configuration empties the cache. The time is given in seconds.

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

Examples:

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

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

Directory Settings

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

- Example: *displayName*
- **2nd User Display Name Field:** An optional second attribute displayed in brackets after the display name, for example using the mail attribute displays as Molly Foo (molly@example.com).
- **Base User Tree:** The base DN of LDAP, from where all users can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one on each line.

• Example:

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

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

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

• Example:

| Server | Users | Log | jin Attributes | Groups | | | Advanced | Expert | | |
|-----------------------------|---------------------|-------------------|------------------|------------------|---|----|-----------|--------|--|--|
| ► Con | Connection Settings | | | | | | | | | |
| - Dire | Directory Settings | | | | | | | | | |
| - Dire | citory Settin | nys | | | | | | | | |
| U | ser Display | Name
Field | displayname | | | | | | | |
| | 2nd User D
Name | isplay
e Field | | | | | | | | |
| | Base User | r Tree | | | | | Base User | Tree | | |
| | | | | | | .: | | _ | | |
| | User S
Attri | earch
ibutes | Optional; one at | tribute per line | 1 | | | | | |
| | | | | | | .: | | | | |
| Gro | oup Display | Name
Field | cn | | | | | | | |
| | Base Group | o Tree | | | | | | | | |
| | | | | | | | | | | |
| | Group S
Attri | earch
ibutes | Optional; one at | tribute per line | | | | | | |
| | | | | | | .: | | | | |
| | Group-Me
asso | ember
ciation | uniqueMember | • | | | | | | |
| Dynamic Group
Member URL | | | | | | | | | | |
| Nested Groups | | | | | | | | | | |
| 1 | Paging chur | nksize | 500 | | | | | | | |
| ▶ Spe | cial Attribu | ites | | | | | | | | |
| Test C | onfiguratio | in i | Help | | | | | | | |

displayName mail

- **Group Display Name Field:** The attribute that should be used as ownCloud group name. ownCloud allows a limited set of characters (a-zA-Z0-9.-_@). Once a group name is assigned it cannot be changed.
 - Example: *cn*
- **Base Group Tree:** The base DN of LDAP, from where all groups can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one in each line.
 - Example:

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

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

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

• Example:

сn description

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

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

• Example: uniquemember

Special Attributes

- **Quota Field:** ownCloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, and it will return human-readable values, e.g. "2 GB". Any quota set in LDAP overrides quotas set on the ownCloud user management page.
 - Example: *ownCloudQuota*
- Quota Default: Override ownCloud default quota for LDAP users who do not have a quota set in the Quota Field.
 - Example: 15 GB
- Email Field: Set the user's email from their LDAP attribute. Leave it empty for default behavior.
 - Example: *mail*

| Server | Users | Login Attributes | Groups | | Advanced | Expert | | | | |
|--------|----------------------|----------------------|--------|--|----------|--------|--|--|--|--|
| + Co | Connection Settings | | | | | | | | | |
| • Dir | Directory Settings | | | | | | | | | |
| - Sp | - Special Attributes | | | | | | | | | |
| | | | | | | | | | | |
| | Qu | ota Field | | | | | | | | |
| | Quota | a Default | | | | | | | | |
| | Em | nail Field | | | | | | | | |
| | User Hom
Nam | e Folder
ing Rule | | | | | | | | |
| Test C | onfiguratio | n ⁱ Help | | | | | | | | |

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

In new ownCloud installations (8.0.10, 8.1.5, 8.2.0 and up) the home folder rule is enforced. This means that once you set a home folder naming rule (get a home folder from an LDAP attribute), it must be available for all users. If it isn't available for a user, then that user will not be able to login. Also, the filesystem will not be set up for that user, so their file shares will not be available to other users.

In existing ownCloud installations the old behavior still applies, which is using the ownCloud username as the home folder when an LDAP attribute is not set. You may change this to enforcing the home folder rule with the occ command in ownCloud 8.2, like this example on Ubuntu:

sudo -u www-data php occ config:app:set user_ldap enforce_home_folder_naming_rule --value=1

6.5.3 Expert Settings

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

Internal Username: The internal username is the identifier in ownCloud for LDAP users. By default it will be created from the UUID attribute. The UUID attribute ensures that the username is unique, and that characters do not need to be converted. Only these characters are allowed: [a-zA-Z0-9_.@-]. Other characters are replaced with their ASCII equivalents, or are simply omitted.

The LDAP backend ensures that there are no duplicate internal usernames in ownCloud, i.e. that it is checking all other activated user backends (including local ownCloud users). On collisions a random number (between 1000 and 9999) will be attached to the retrieved value. For example, if "alice" exists, the next username may be "alice_1337".

| Server | Users | Login Attributes | Groups | | Advanced | Expert | | |
|----------|--|----------------------------|-----------------|--|-------------------|----------|--|--|
| Internal | Internal Username | | | | | | | |
| I | By default the internal username will be created from the UUID attribute. It makes sure that the username is unique and | | | | | | | |
| 0 | characters do not need to be converted. The internal username has the restriction that only these characters are allowed: [| | | | | | | |
| á | a-zA-Z0-9(| @-]. Other characters | are replaced | with their ASCII correspondence or simply omitte | d. On collisions | a | | |
| 1 | number will | be added/increased. T | ne internal us | emame is used to identify a user internally. It is a | also the default | name for | | |
| 1 | the user hon | ne folder. It is also a pa | art of remote | URLs, for instance for all *DAV services. With thi | s setting, the de | efault | | |
| | behavior can | i be overridden. To ach | ieve a similar | behavior as before ownCloud 5 enter the user d | splay name atti | ibute in | | |
| | the following | Tield. Leave it empty i | or default be | navior. Changes will have effect only on newly ma | apped (added) l | DAP | | |
| | users. | | | | | | | |
| | Interna | al Username | | | | | | |
| 0 | | Attribute: | | | | | | |
| Override | OUID deter | cuon | tomotionlluid | started. The ULUD attribute is used to doubtleed | u identifu LDAE | | | |
| | by default, t | Also, the internal user | nomatically u | created based on the LILID, if not specified other | wise above. Yo | users | | |
| | override the | setting and pass an at | tribute of you | r choice. You must make sure that the attribute of | f your choice ca | an be | | |
| 1 | etched for b | oth users and groups a | and it is uniqu | e. Leave it empty for default behavior. Changes | will have effect | only on | | |
| | newly mappe | ed (added) LDAP users | and groups. | | | | | |
| | UUID | Attribute for | | | | | | |
| | | Users: | | | | | | |
| | UUID | Attribute for | | | | | | |
| | | Groups: | | | | | | |
| Usernam | e-LDAP Us | er Mapping | | | | | | |
| L. L. | Usernames a | are used to store and a | issign (meta) | data. In order to precisely identify and recognize | users, each LD. | AP user | | |
| 1 | will have an | internal username. Thi | s requires a r | napping from username to LDAP user. The creat | ed username is | mapped | | |
| 1 | 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 lettovers everywhere. Clearing the mappings is not contiguration 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 | | | | | | | |
| | | | | | | | | |
| Test C | onfiguratio | n ⁱ Help | | | | | | |

The internal username is the default name for the user home folder in ownCloud. It is also a part of remote URLs, for instance for all *DAV services.

You can override all of this with the Internal Username setting. Leave it empty for default behaviour. Changes will affect only newly mapped LDAP users.

- Example: uid
- **Override UUID detection** By default, ownCloud auto-detects the UUID attribute. The UUID attribute is used to uniquely identify LDAP users and groups. The internal username will be created based on the UUID, if not specified otherwise.

You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behaviour. Changes will have effect only on newly mapped LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached, which will result in a new user. Because of this, the setting should be applied before putting ownCloud in production use and clearing the bindings (see the User and Group Mapping section below).

- Example: *cn*
- **Username-LDAP User Mapping** ownCloud uses usernames as keys to store and assign data. In order to precisely identify and recognize users, each LDAP user will have a internal username in ownCloud. This requires a mapping from ownCloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by ownCloud by checking the UUID value.

The same is valid for groups.

The internal ownCloud name is used all over in ownCloud. Clearing the Mappings will have leftovers everywhere. Never clear the mappings in a production environment, but only in a testing or experimental server.

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

6.5.4 Testing the configuration

The **Test Configuration** button checks the values as currently given in the input fields. You do not need to save before testing. By clicking on the button, ownCloud will try to bind to the ownCloud server using the settings currently given in the input fields. If the binding fails you'll see a yellow banner with the error message "The configuration is invalid. Please have a look at the logs for further details."

When the configuration test reports success, save your settings and check if the users and groups are fetched correctly on the Users page.

6.5.5 ownCloud Avatar integration

ownCloud supports user profile pictures, which are also called avatars. If a user has a photo stored in the *jpegPhoto* or *thumbnailPhoto* attribute on your LDAP server, it will be used as their avatar. In this case the user cannot alter their avatar (on their Personal page) as it must be changed in LDAP. *jpegPhoto* is preferred over *thumbnailPhoto*.

If the *jpegPhoto* or *thumbnailPhoto* attribute is not set or empty, then users can upload and manage their avatars on their ownCloud Personal pages. Avatars managed in ownCloud are not stored in LDAP.

The *jpegPhoto* or *thumbnailPhoto* attribute is fetched once a day to make sure the current photo from LDAP is used in ownCloud. LDAP avatars override ownCloud avatars, and when an LDAP avatar is deleted then the most recent ownCloud avatar replaces it.

Photos served from LDAP are automatically cropped and resized in ownCloud. This affects only the presentation, and the original image is not changed.



Your avatar is provided by your original account.

6.5.6 Troubleshooting, Tips and Tricks

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
- Using LDAPS, also make sure that the port is correctly configured (by default 636)

Microsoft Active Directory

Compared to earlier ownCloud versions, no further tweaks need to be done to make ownCloud work with Active Directory. ownCloud will automatically find the correct configuration in the set-up process.

memberOf / Read MemberOf permissions

If you want to use memberOf within your filter you might need to give your querying user the permissions to use it. For Microsoft Active Directory this is described here.

Duplicating Server Configurations

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

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

Now you can modify and enable the configuration.

6.5.7 Performance tips

See the documentation wiki for additional LDAP tips and tricks. The following performance tips are standard for inmproving LDAP performance.

Caching

Using caching to speed up lookups. See *Configuring Memory Caching*). The ownCloud cache is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

You can adjust the **Cache Time-To-Live** value to balance performance and freshness of LDAP data. All LDAP requests will be cached for 10 minutes by default, and you can alter this with the **Cache Time-To-Live** setting. The cache answers each request that is identical to a previous request, within the time-to-live of the original request, rather than hitting the LDAP server.

The **Cache Time-To-Live** is related to each single request. After a cache entry expires there is no automatic trigger for re-populating the information, as the cache is populated only by new requests, for example by opening the User administration page, or searching in a sharing dialog.

There is one trigger which is automatically triggered by a certain background job which keeps the user-group-mappings up-to-date, and always in cache.

Under normal circumstances, all users are never loaded at the same time. Typically the loading of users happens while page results are generated, in steps of 30 until the limit is reached or no results are left. For this to work on an oC-Server and LDAP-Server, **Paged Results** must be supported, which presumes PHP \ge 5.4.

ownCloud remembers which user belongs to which LDAP-configuration. That means each request will always be directed to the right server unless a user is defunct, for example due to a server migration or unreachable server. In this case the other servers will also receive the request.

LDAP indexing

Turn on indexing. Deciding which attributes to index depends on your configuration, and which LDAP server you are using. See openLDAP Indexes for openLDAP, and How to Index an Attribute in Active Directory for Active Directory. The openLDAP how to is especially useful for figuring out which attributes to index.

Use precise base DNs

The more precise your base DN, the faster LDAP can search because it has fewer branches to search.

Use precise filters

Use good filters to further define the scope of LDAP searches, and to intelligently direct your server where to search, rather than forcing it to perform needlessly-general searches.

6.5.8 ownCloud LDAP Internals

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

User and Group Mapping

In ownCloud the user or group name is used to have all relevant information in the database assigned. To work reliably a permanent internal user name and group name is created and mapped to the LDAP DN and UUID. If the DN changes in LDAP it will be detected, and there will be no conflicts.

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

As of ownCloud 5 the internal user name and a visible display name are separated. This is not the case for group names, yet, i.e. a group name cannot be altered.

That means that your LDAP configuration should be good and ready before putting it into production. The mapping tables are filled early, but as long as you are testing, you can empty the tables any time. Do not do this in production.

Handling with Backup Server

When ownCloud is not able to contact the main LDAP server, ownCloud assumes it is offline and will not try to connect again for the time specified in **Cache Time-To-Live**. If you have a backup server configured ownCloud will connect to it instead. When you have scheduled downtime, check **Disable Main Server** to avoid unnecessary connection attempts.

6.6 LDAP User Cleanup

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

These items are removed upon cleanup:

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

There are two prerequisites for LDAP User Cleanup to operate:

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

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

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

These examples are for Ubuntu Linux:

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

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

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

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

6.6.1 Deleting Local ownCloud Users

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

6.7 User Provisioning API

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

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

6.7.1 Instruction Set For Users

users / adduser

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

Syntax: ocs/v1.php/cloud/users

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

Status codes:

• 100 - successful

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

Example

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

XML Output

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

users / getusers

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

Syntax: ocs/v1.php/cloud/users

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

Status codes:

• 100 - successful

Example

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

XML Output

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

users / getuser

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

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

• HTTP method: GET

Status codes:

• 100 - successful

Example

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

XML Output

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

users / edituser

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

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

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

• PUT argument: value, the new value for the field

Status codes:

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

Examples

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

XML Output

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

users / deleteuser

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

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

• HTTP method: DELETE

Statuscodes:

- 100 successful
- 101 failure

Example

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

XML Output

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

users / getgroups

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

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

• HTTP method: GET

Status codes:

• 100 - successful

Example

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

XML Output

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

users / addtogroup

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

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

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

Status codes:

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

Example

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

XML Output

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

users / removefromgroup

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

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

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

Status codes:

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

Example

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

XML Output

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

users / createsubadmin

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

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

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

Status codes:

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

Example

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

XML Output

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

users / removesubadmin

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

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

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

Status codes:

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

Example

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

XML Output

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

users / getsubadmingroups

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

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

• HTTP method: GET

Status codes:

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

Example

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

XML Output

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

6.7.2 Instruction Set For Groups

groups / getgroups

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

Syntax: ocs/v1.php/cloud/groups

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

Status codes:

• 100 - successful

Example

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

XML Output

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

```
<data>
<groups>
<element>admin</element>
</groups>
</data>
</ocs>
```

groups / addgroup

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

Syntax: ocs/v1.php/cloud/groups

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

Status codes:

- 100 successful
- 101 invalid input data
- 102 group already exists
- 103 failed to add the group

Example

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

XML Output

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

groups / getgroup

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

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

• HTTP method: GET

Status codes:

• 100 - successful

Example

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

XML Output

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

groups / getsubadmins

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

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

• HTTP method: GET

Status codes:

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

Example

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

XML Output

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

</data> </ocs>

groups / deletegroup

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

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

• HTTP method: DELETE

Status codes:

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

Example

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

XML Output

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

6.7.3 Instruction Set For Apps

apps / getapps

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

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

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

Status codes:

- 100 successful
- 101 invalid input data

Example

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

XML Output

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

apps / getappinfo

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

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

• HTTP method: GET

Status codes:

• 100 - successful

Example

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

XML Output

```
<?rml version="1.0"?>
<ocs>
<meta>
<statuscode>100</statuscode>
<status>ok</status>
</meta>
<data>
<info/>
<remote>
<files>appinfo/remote.php</files>
<webdav>appinfo/remote.php</webdav>
<filesync>appinfo/filesync.php</filesync>
</remote>
```

```
<public/>
<id>files</id>
</description>File Management</description>
<licence>AGPL</licence>
<author>Robin Appelman</author>
<require>4.9</require>
<shipped>true</shipped>
<standalone></standalone>
<default_enable></default_enable>
<types>
<element>filesystem</element>
</types>
</data>
</ocs>
```

apps / enable

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

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

• HTTP method: POST

Status codes:

• 100 - successful

Example

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

XML Output

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

apps / disable

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

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

• HTTP method: DELETE

Status codes:

• 100 - successful

Example

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

XML Output

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

CHAPTER

SEVEN

FILE SHARING AND MANAGEMENT

7.1 File Sharing

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

- Allow users to share files
- Allow users to create public shares
- Require a password on public shares
- · Allow public uploads to public shares
- Require an expiration date on public share links
- Allow resharing
- Restrict sharing to group members only
- Allow email notifications of new public shares
- · Exclude groups from creating shares

Note: ownCloud Enterprise includes a Share Link Password Policy app; see Share Link Password Policy.

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

- Check Allow apps to use the Share API to enable users to share files. If this is not checked, no users can create file shares.
- Check Allow users to share via link to enable creating public shares for people who are not own-Cloud users via hyperlink.
- Check Enforce password protection to force users to set a password on all public share links. This does not apply to local user and group shares.
- Check Allow public uploads to allow anyone to upload files to public shares.
- Check Allow users to send mail notification for shared files to enable sending notifications from ownCloud. (Your ownCloud server must be configured to send mail)
- Check Set default expiration date to set a default expiration date on public shares.
- Check Allow resharing to enable users to re-share files shared with them.
- Check Restrict users to only share with users in their groups to confine sharing within group memberships.

Sharing *i*

Allow apps to use the Share API

- 🕑 Allow users to share via link
 - Enforce password protection
 - Allow public uploads
 - Allow users to send mail notification for shared files
 - Set default expiration date
- Allow resharing
- Restrict users to only share with users in their groups
- Allow users to send mail notification for shared files to other users
- Exclude groups from sharing

Allow username autocompletion in share dialog. If this is disabled the full username needs to be entered.

Note: This setting does not apply to the Federated Cloud sharing feature. If *Federated Cloud Sharing* is enabled, users can still share items with any users on any instances (including the one they are on) via a remote share.

- Check Allow users to send mail notification for shared files enables users to send an email notification to every ownCloud user that the file is shared with.
- Check Exclude groups from sharing to prevent members of specific groups from creating any file shares in those groups. When you check this, you'll get a dropdown list of all your groups to choose from. Members of excluded groups can still receive shares, but not create any
- Check Allow username autocompletion in share dialog to enable auto-completion of own-Cloud usernames.

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

7.1.1 Transferring Files to Another User

You may transfer files from one user to another with occ. This is useful when you have to remove a user. Be sure to transfer the files before you delete the user! This transfers all files from user1 to user2, and the shares and metadata info associated with those files (shares, tags, comments, etc). Trashbin contents are not transferred:

occ files:transfer-ownership user1 user2

(See *Using the occ Command* for a complete occ reference.)
7.1.2 Creating Persistent File Shares

When a user is deleted, their files are also deleted. As you can imagine, this is a problem if they created file shares that need to be preserved, because these disappear as well. In ownCloud files are tied to their owners, so whatever happens to the file owner also happens to the files.

One solution is to create persistent shares for your users. You can retain ownership of them, or you could create a special user for the purpose of establishing permanent file shares. Simply create a shared folder in the usual way, and share it with the users or groups who need to use it. Set the appropriate permissions on it, and then no matter which users come and go, the file shares will remain. Because all files added to the share, or edited in it, automatically become owned by the owner of the share regardless of who adds or edits them.

7.1.3 Share Link Password Policy

ownCloud Enterprise users have the option of enabling the Share Link Password Policy app. This allows you to enforce password length, required characters, define special characters, and expiration dates on share links.

Share li	nk password policy
Passwords	should have at least:
8	minimum characters
□ 1	uppercase letters
□ 1	numbers
□ 1	special characters
🗌 Def	fine special characters #!
Link expirat	tion:
7	days to expire link if password is set
□ 7	days to expire link if password is not set
Save	

Note that you cannot use Emojis as special characters with MySQL, as it supports UTF8 characters only of 1-3 bytes, and emojis require 4 bytes.

7.2 Configuring Federation Sharing

Federated Cloud Sharing is now managed by the Federation app (9.0+), and is now called Federation sharing. When you enable the Federation app you can easily and securely link file shares between ownCloud servers, in effect creating a cloud of ownClouds.

7.2.1 Sharing With ownCloud 8 and Older

Direct Federation shares (*Creating a new Federation Share* (9.0+ only)) are not supported in ownCloud 8 and older, so you must create Federation shares with public links (*Creating Federation Shares via Public Link Share*).

7.2.2 Creating a new Federation Share (9.0+ only)

Follow these steps to create a new Federation share between two ownCloud 9.0+ servers. This requires no action by the user on the remote server; all it takes is a few steps on the originating server.

- 1. Enable the Federation app.
- 2. Go to your ownCloud Admin page and scroll to the Sharing section. Verify that Allow users on this server to send shares to other servers and Allow users on this server to receive shares from other servers are enabled.
- 3. Now go to the Federation section. By default, **Add server automatically once a federated share was created successfully** is checked. The Federation app supports creating a list of trusted ownCloud servers, which allows the trusted servers to exchange user directories and auto-complete the names of external users when you create shares. If you do not want this enabled, then un-check it.

Federation

ownCloud Federation allows you to connect with other trusted ownClouds to exchange the user directory. For example this will be used to auto-complete external users for federated sharing.

Add server automatically once a federated share was created successfully

Trusted ownCloud Servers

+ Add ownCloud server

- 3. Now go to your Files page and select a folder to share. Click the share icon, and then enter the username and URL of the user on the remote ownCloud server. In this example, that is freda@https://example.com/owncloud. When ownCloud verifies the link, it displays it with the (remote) label. Click on this label to establish the link.
- 3. When the link is successfully completed, you have a single share option, and that is can edit.

You may disconnect the share at any time by clicking the trash can icon.





7.2.3 Configuring Trusted ownCloud Servers

You may create a list of trusted ownCloud servers for Federation sharing. This allows your linked ownCloud servers to share user directories, and to auto-fill user names in share dialogs. If **Add server automatically once a federated share was created successfully** is enabled on your Admin page, servers will be automatically added to your trusted list when you create new Federation shares.

You may also enter ownCloud server URLs in the **Add ownCloud Server** field. The yellow light indicates a successful connection, with no user names exchanged. The green light indicates a successful connection with user names exchanged. A red light means the connection failed.



7.2.4 Creating Federation Shares via Public Link Share

You'll need to use a Public Link Share to create Federation shares with ownCloud 8.x and older.

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

☑ Share link	
http://ubuntu2/owncloud/index.php/s/	ujh7VbEFuvjXIGu
Password protect	
friend@example.com	Send
Set expiration date	

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

DownCloud		Add to your ownCloud	↓ Download
	↓ Download nebula-3.jpg (7 kB)		
Direct link	http://ubuntu2/owncloud/index.php/s/OA1w	CoQI	

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

	note-server/owncloud	+ Download
and the second of the		

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

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

7.2.5 Configuration Tips

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

ġ	Files 🔻	٩		layla 🔻
	All files			
*	Favorites	□ Name 🛦	Size	Modified
\$	Shared with you	Documents	35 kB	34 minutes ago
\$	Shared with others	Photos	663 kB	34 minutes ago
S	Shared by link	ownCloudUserManual.pdf	1.9 MB	34 minutes ago
		Remote share	×	
		Do you want to add the remote share nebula-3.jpg from admin@ubuntu2/ownclou	ıd?	
		Cancel Add rem	ote share	

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

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

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

7.3 Uploading big files > 512MB

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

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

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

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

7.3.1 System Configuration

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

7.3.2 Configuring Your Web server

Note: ownCloud comes with its own owncloud/.htaccess file. Because php-fpm can't read PHP settings in .htaccess these settings must be set in the owncloud/.user.ini file.

Set the following two parameters inside the corresponding php.ini file (see the **Loaded Configuration File** section of *PHP Version and Information* to find your relevant php.ini files)

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

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

php_value max_input_time 3600
php_value max_execution_time 3600

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

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

Apache

- LimitRequestBody
- SSLRenegBufferSize

Apache with mod_fcgid

- FcgidMaxRequestInMem
- FcgidMaxRequestLen

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

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

nginx

- client_max_body_size
- fastcgi_read_timeout
- client_body_temp_path

Since nginx 1.7.11 a new config option fastcgi_request_buffering is availabe. Setting this option to fastcgi_request_buffering off; in your nginx config might help with timeouts during the upload. Furthermore it helps if you're running out of disc space on the tmp partition of your system.

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

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

If your site is behind a nginx frontend (for example a loadbalancer):

By default, downloads will be limited to 1GB due to proxy_buffering and proxy_max_temp_file_size on the frontend.

- If you can access the frontend's configuration, disable proxy_buffering or increase proxy_max_temp_file_size from the default 1GB.
- If you do not have access to the frontend, set the X-Accel-Buffering header to add_header X-Accel-Buffering no; on your backend server.

7.3.3 Configuring PHP

If you don't want to use the ownCloud .htaccess or .user.ini file, you may configure PHP instead. Make sure to comment out any lines .htaccess pertaining to upload size, if you entered any.

If you are running ownCloud on a 32-bit system, any open_basedir directive in your php.ini file needs to be commented out.

Set the following two parameters inside php.ini, using your own desired file size values:

upload_max_filesize = 16G
post_max_size = 16G

Tell PHP which temp file you want it to use:

upload_tmp_dir = /var/big_temp_file/

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

• output_buffering = 0

7.3.4 Configuring ownCloud

As an alternative to the upload_tmp_dir of PHP (e.g. if you don't have access to your php.ini) you can also configure a temporary location for uploaded files by using the tempdirectory setting in your config.php (See *Config.php Parameters*).

If you have configured the session_lifetime setting in your config.php (See *Config.php Parameters*) file then make sure it is not too low. This setting needs to be configured to at least the time (in seconds) that the longest

upload will take. If unsure remove this completely from your configuration to reset it to the default shown in the config.sample.php.

Configuring upload limits within the GUI

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

File handling		
Maximum upload size 1 GB		
With PHP-FPM this value may	take up to 5 minute	es to take effect after saving.
Save		
Depending on your environment you might ge	t an insufficient permissions	message shown for this input box.
File handling		
Maximum upload size	1 GB	

Can not be edited from here due to insufficient permissions.

To be able to use this input box you need to make sure that:

- your Web server is be able to use the .htaccess file shipped by ownCloud (Apache only)
- the user your Web server is running as has write permissions to the files .htaccess and .user.ini

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

7.3.5 General upload issues

Various environmental factors could cause a restriction of the upload size. Examples are:

- the LVE Manager of CloudLinux which sets a I/O limit
- some services like Cloudflare are also known to cause uploading issues
- · upload limits enforced by proxies used by your clients
- other webserver modules like described in General Troubleshooting

7.4 Configuring the Collaborative Documents App

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

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

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

7.4.1 Enabling the Documents App

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

Documents 0.8.2 Internal App

An ownCloud app to work with office documents

AGPL-licensed by Frank Karlitschek

Disable

🗹 Enable only for specific groups



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

7.4.2 Enabling and testing MS Word support

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

Documents

MS Word support (requires openOffice/libreOffice)

Local

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

○ External

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

Disabled

No MS Word support

Apply and test Saved

Troubleshooting

If the mentioned test fails please make sure that:

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

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

7.5 Providing Default Files

You may distribute a set of default files and folders to all users by placing them in the <code>owncloud/core/skeleton</code> directory on your ownCloud server. These files appear only to new users after their initial login, and existing users will not see files that are added to this directory after their first login. The files in the <code>skeleton</code> directory are copied into the users' data directories, so they may change and delete the files without affecting the originals.

This screenshot shows a set of photos in the skeleton directory.



They appear on the user's ownCloud Files page just like any other files.

Kiles 🔹		٩	molly v
All files	# > Photos > New 1		
Favorites	🔍 Name 🔺	Size	Modified
Shared with you	annabelle.png	2.3 MB	6 minutes ago
Shared with others	beerstuff.jpeg	177 kB	6 minutes ago
	layla-cinder-snow.png	5.3 MB	6 minutes ago
	no-fish-cleaning.jpeg	260 kB	6 minutes ago
	Paris.jpg	223 kB	6 minutes ago
	San Francisco.jpg	211 kB	6 minutes ago
	Squirrel.jpg	228 kB	6 minutes ago
Deleted files	7 files	8.6 MB	

7.5.1 Additional Configuration

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

7.6 Configuring External Storage (GUI)

The External Storage Support application enables you to mount external storage services and devices as secondary ownCloud storage devices. You may also allow users to mount their own external storage services.

ownCloud 9.0 introduces a new set of occ commands for managing external storage.

Also new in 9.0 is an option for the ownCloud admin to enable or disable sharing on individual external mountpoints (see *Mount Options*). Sharing on such mountpoints is disabled by default.

7.6.1 Enabling External Storage Support

Warning: Enabling this app will disable the Stay logged in checkbox on the login page.

The External storage support application is enabled on your Apps page.



7.6.2 Storage Configuration

To create a new external storage mount, select an available backend from the dropdown **Add storage**. Each backend has different required options, which are configured in the configuration fields.

External Storage			
Folder name	External storage	Configuration	Available for
Folder name	Add storage 🔹		
	Amazon S3 and compli Dropbox FTP	ant	
	Google Drive Local OpenStack Object Sto ownCloud SFTP SMB / CIFS SMB / CIFS using OC lo WebDAV	rage igin	

Each backend may also accept multiple authentication methods. These are selected with the dropdown under **Authentication**. Different backends support different authentication mechanisms; some specific to the backend, others are more generic. See *External Storage Authentication mechanisms* for more detailed information.

When you select an authentication mechanism, the configuration fields change as appropriate for the mechanism. The SFTP backend, for one example, supports **username and password**, **Log-in credentials**, **save in session**, and **RSA public key**.

Required fields are marked with a red border. When all required fields are filled, the storage is automatically saved. A green dot next to the storage row indicates the storage is ready for use. A red or yellow icon indicates that ownCloud could not connect to the external storage, so you need to re-check your configuration and network availability.

If there is an error on the storage, it will be marked as unavailable for ten minutes. To re-check it, click the colored icon or reload your Admin page.

Folder name	External storage	Authentication	Configuration
			Host
CETD	CETD	Username and password	Root
STIF	5111	Username and password	Username
		Log-in credentials, save in session RSA public key	Password

External Storage

7.6.3 User and Group Permissions

A storage configured in a user's Personal settings is available only to the user that created it. A storage configured in the Admin settings is available to all users by default, and it can be restricted to specific users and groups in the **Available for** field.



7.6.4 Mount Options

Hover your cursor to the right of any storage configuration to expose the settings button and trashcan. Click the trashcan to delete the mountpoint. The settings button allows you to configure each storage mount individually with the following options:

- Encryption
- Previews
- Enable Sharing
- Filesystem check frequency (Never, Once per direct access)

The Encryption checkbox is visible only when the Encryption app is enabled.

Enable Sharing allows the ownCloud admin to enable or disable sharing on individual mountpoints. When sharing is disabled the shares are retained internally, so that you can re-enable sharing and the previous shares become available again. Sharing is disabled by default.

All users. Type to se	÷	١.			
 Enable previews Enable sharing 					
Check for changes Once every direct access					
Never					
	Once every direct acc	ess			

7.6.5 Using Self-Signed Certificates

When using self-signed certificates for external storage mounts the certificate must be imported into ownCloud. Please refer to *Importing System-wide and Personal SSL Certificates* for more information.

7.6.6 Available storage backends

The following backends are provided by the external storages app. Other apps may provide their own backends, which are not listed here.

Amazon S3

To connect your Amazon S3 buckets to ownCloud, you will need:

- S3 access key
- S3 secret key
- Bucket name

In the Folder name field enter a local folder name for your S3 mountpoint. If this does not exist it will be created.

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

The Enable SSL checkbox enables HTTPS connections; using HTTPS is always highly-recommended.

Optionally, you can override the hostname, port and region of your S3 server, which is required for non-Amazon servers such as Ceph Object Gateway.

Enable path style is usually not required (and is, in fact, incompatible with newer Amazon datacenters), but can be used with non-Amazon servers where the DNS infrastructure cannot be controlled. Ordinarily, requests will be made with http://bucket.hostname.domain/, but with path style enabled, requests are made with http://hostname.domain/bucket instead.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.



External Storage

Dropbox

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

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



Next, for Choose an API check Dropbox API.

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

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

1. Choose an API



2. Choose the type of access you need

Learn more about access types

• App folder – Access to a single folder created specifically for your app.

• Full Dropbox – Access to all files and folders in a user's Dropbox.

3. Name your app

oc-share

Then click the Create App button.

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

oc-share				
Settings	Branding	Analytics		
Status		Development	Apply for	production
Development	users	Only you	Enable addi	tional users
Permission typ	e	Full Dropbox 🜖		
App key App secret		Show		

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

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

After entering your local mount name, App Key and App Secret, click Grant access.

				mb4b0fg4rbi2mmv			
•	Dropbox2	Dropbox	OAuth1 🝷	*******	All users. Type to select user or group.	o	Ŵ
				Grant access			
				Access granted			

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

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

FTP/FTPS

To connect to an FTP server, you will need:

- A folder name for your local mountpoint; the folder will be created if it does not exist
- The URL of the FTP server
- Port number (default: 21)



- FTP server username and password
- Remote Subfolder, the FTP directory to mount in ownCloud. ownCloud defaults to the root directory. If you specify a subfolder you must leave off the leading slash. For example, public_html/images

Your new mountpoint is available to all users by default, and you may restrict access by entering specific users or groups in the **Available for** field.

Optionally, ownCloud can use FTPS (FTP over SSL) by checking **Secure ftps://**. This requires additional configuration with your root certificate if the FTP server uses a self-signed certificate (See *Importing System-wide and Personal SSL Certificates*).

External Storage

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

Note: The external storage FTP/FTPS needs the allow_url_fopen PHP setting to be set to 1. When having connection problems make sure that it is not set to 0 in your php.ini. See *PHP Version and Information* to learn how to find the right php.ini file to edit.

See Configuring External Storage (GUI) for additional mount options and information.

FTP uses the password authentication scheme; see *External Storage Authentication mechanisms* for more information on authentication schemes.

Google Drive

ownCloud uses OAuth 2.0 to connect to Google Drive. This requires configuration through Google to get an app ID and app secret, as ownCloud registers itself as an app.

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

If you already have a Google account, such as Groups, Drive, or Mail, you can use your existing login to log into the Google Cloud Console. After logging in click the **Create Project** button.

		٩	ocSync 🔻
	Allprojects myownCloudApp ✓ ocSync ProjectStor	myowncloudapp loyal-glass-719 calcium-vial-719	
#3	Manage all projects Create a project		

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

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

You'll be returned to your dashboard.

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

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

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

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





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



Overview



Drive API

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

Go to Credentials

Credentials

Credentials OAuth consent screen Domain verification



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

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

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

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

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

Credentials

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

Create Cancel

OAuth clie	nt		
Here is your	client ID		
0		.apps.googleusercontent.c	om
Here is your	client secret		
80Ht	S. Contente		

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

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

😑 Google Develope	rs Console Q gdrive-owncloud - 👬 🖪 🕫 🤨 🗄		
API API Manager	Credentials		
 ♦ Overview 	Credentials OAuth consent screen Domain verification		
O- Credentials			
	Add credentials 👻 Delete		
	Create credentials to access your enabled APIs. Refer to the API documentation for details.		
	OAuth 2.0 client IDs		
	□ Name Creation date ~ Type Client ID		
	MyGoogleDriveApp Dec 1, 2015 Web 6000000000000000000000000000000000000		

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

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

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

See Configuring External Storage (GUI) for additional mount options and information.

See *External Storage Authentication mechanisms* for more information on authentication schemes. 603026686136qnv90oocacrkrh1vs0cht83eprgm2sbb.apps.googleusercontent.com

MyGoogleDriveApp would like to: View and manage the files in your Google Drive Image the files in your Google Drive Image the files app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.	
View and manage the files in your Google Drive (i) By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.	
By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.	
Deny Allow	
External Storage	
Folder name External storage Authentication Configuration	1
6000000000	- nvs
Google Drive OAuth2 -	
Grant acces	s
Access grante	ed

Local

Local storages provide access to any directory on the ownCloud server. Since this is a significant security risk, Local storage can only be configured in the ownCloud admin settings. Non-admin users cannot create Local storage mounts.

Use this to mount any directory on your ownCloud server that is outside of your ownCloud data/ directory. This directory must be readable and writable by your HTTP server user. These ownership and permission examples are on Ubuntu Linux:

```
sudo -u www-data chown -R www-data:www-data /localdir
sudo -u www-data chmod -R 0750 /localdir
```

See *Setting Strong Directory Permissions* for information on correct file permissions, and find your HTTP user *PHP Version and Information*.

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

In the **Configuration** field enter the full filepath of the directory you want to mount.

In the **Available for** field enter the users or groups who have permission to access the mount. By default all users have access.

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

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

OpenStack Object Storage

OpenStack Object Storage is used to connect to an OpenStack Swift server, or to Rackspace. Two authentication mechanisms are available: one is the generic OpenStack mechanism, and the other is used exclusively for Rackspace, a provider of object storage that uses the OpenStack Swift protocol.

The OpenStack authentication mechanism uses the OpenStack Keystone v2 protocol. Your ownCloud configuration needs:

- **Bucket**. This is user-defined; think of it as a subdirectory of your total storage. The bucket will be created if it does not exist.
- Username of your account.
- Password of your account.
- Tenant name of your account. (A tenant is similar to a user group.)
- Identity Endpoint URL, the URL to log in to your OpenStack account.

The Rackspace authentication mechanism requires:

- Bucket
- Username
- API key.



You must also enter the term **cloudFiles** in the **Service name** field.



It may be necessary to specify a **Region**. Your region should be named in your account information, and you can read about Rackspace regions at About Regions.

The timeout of HTTP requests is set in the **Request timeout** field, in seconds.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

ownCloud

An ownCloud storage is a specialized *WebDAV* storage, with optimizations for ownCloud-ownCloud communication. See the *WebDAV* documentation to learn how to configure an ownCloud external storage.

When filling in the URL field, use the path to the root of the ownCloud installation, rather than the path to the WebDAV endpoint. So, for a server at https://example.com/owncloud, use https://example.com/owncloud and not https://example.com/owncloud/remote.php/dav.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

SFTP

ownCloud's SFTP (FTP over an SSH tunne) backend supports both password and public key authentication.

The Host field is required; a port can be specified as part of the Host field in the following format: hostname.domain:port. The default port is 22 (SSH).

For public key authentication, you can generate a public/private key pair from your SFTP with secret key login configuration.

External Storage

Folder name	External storage	Authentication	Configuration
			Host
CETD	CETD		Root
SEIF	2616	Username and password	Username
		Log-in credentials, save in session RSA public key	Password

After generating your keys, you need to copy your new public key to the destination server to .ssh/authorized_keys.ownCloud will then use its private key to authenticate to the SFTP server.

The default **Remote Subfolder** is the root directory (/) of the remote SFTP server, and you may enter any directory you wish.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

SMB/CIFS

ownCloud can connect to Windows file servers or other SMB-compatible servers with the SMB/CIFS backend.

Note: The SMB/CIFS backend requires the PHP smbclient module to be installed on the ownCloud server. This should be included in any Linux distribution; on Debian, Ubuntu, CentOS, and Fedora it is php-smbclient. See eduardok/libsmbclient-php if your distribution does not include it; this provides source archives and instructions how to install binary packages.

You also need the Samba client installed on your Linux system. This is included in all Linux distributions; on Debian, Ubuntu, and other Debian derivatives this is smbclient. On SUSE, Red Hat, CentOS, and other Red Hat derivatives it is samba-client. You also need which and stdbuf, which should be included in most Linux distributions.

You need the following information:

- Folder name for your local mountpoint.
- Host: The URL of the Samba server.
- Username: The username or domain/username used to login to the Samba server.
- Password: the password to login to the Samba server.
- Share: The share on the Samba server to mount.
- Remote Subfolder: The remote subfolder inside the Samba share to mount (optional, defaults to /). To assign the ownCloud logon username automatically to the subfolder, use *suser* instead of a particular subfolder name.
- And finally, the ownCloud users and groups who get access to the share.

Optionally, you can specify a Domain. This is useful in cases where the SMB server requires a domain and a username, and an advanced authentication mechanism like session credentials is used so that the username cannot be modified. This is concatenated with the username, so the backend gets domain/username

			smbserver	
smbcifs	SMB / CIES	Session credentials	users	All users. Type to select
Shibelis	Sind / Cirb	Username and password	/shared	
		Session credentials	Domain	

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

WebDAV

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

You need the following information:

- Folder name: The name of your local mountpoint.
- The URL of the WebDAV or ownCloud server.
- · Username and password for the remote server
- Secure https://: We always recommend https:// for security, though you can leave this unchecked for http://.

Optionally, a Remote Subfolder can be specified to change the destination directory. The default is to use the whole root.

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

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

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

			https://remoteserve	
			admin	
•	oc-remote	ownCloud		All Users ×
			4	
			Secure https://	

7.6.7 Allow Users to Mount External Storage

Check **Enable User External Storage** to allow your users to mount their own external storage services, and check the backends you want to allow. Beware, as this allows a user to make potentially arbitrary connections to other services on your network!

~	Enable User External Storage		
	Allow users to mount the following external storage		
	 Amazon S3 and compliant 		
	✓ Dropbox		
	FTP		
	Google Drive		
	OpenStack Object Storage		
	ownCloud		
	SFTP		
	SMB / CIFS		
	SMB / CIFS using OC login		
	WebDAV		

7.6.8 Adding Files to External Storages

We recommend configuring the background job **Webcron** or **Cron** (see *Background Jobs*) to enable ownCloud to automatically detect files added to your external storages.

Note: You cannot scan/detect changed files on external storage mounts when you select the **Log-in credentials**, **save in session** authentication mechanism. However, there is a workaround, and that is to use Ajax cron mode. See *Password-based Mechanisms* for more information.

ownCloud may not always be able to find out what has been changed remotely (files changed without going through ownCloud), especially when it's very deep in the folder hierarchy of the external storage.

You might need to setup a cron job that runs sudo -u www-data php occ files:scan --all (or replace "-all" with the user name, see also *Using the occ Command*) to trigger a rescan of the user's files periodically (for example every 15 minutes), which includes the mounted external storage.

7.7 Configuring External Storage (Configuration File)

Starting with ownCloud 9.0, the data/mount.json file for configuring external storages has been removed, and replaced with a set of *occ commands*.

7.8 External Storage Authentication mechanisms

ownCloud storage backends accept one or more authentication schemes such as passwords, OAuth, or token-based, to name a few examples. Each authentication scheme may be implemented by multiple authentication mechanisms. Different mechanisms require different configuration parameters, depending on their behaviour.

7.8.1 Special Mechanisms

The **None** authentication mechanism requires no configuration parameters, and is used when a backend requires no authentication.

The **Built-in** authentication mechanism itself requires no configuration parameters, but is used as a placeholder for legacy storages that have not been migrated to the new system and do not take advantage of generic authentication mechanisms. The authentication parameters are provided directly by the backend.

7.8.2 Password-based Mechanisms

The **Username and password** mechanism requires a manually-defined username and password. These get passed directly to the backend.

The **Log-in credentials, save in session** mechanism uses the ownCloud login credentials of the user to connect to the storage. These are not stored anywhere on the server, but rather in the user session, giving increased security. The drawbacks are that sharing is disabled when this mechanism is in use, as ownCloud has no access to the storage credentials, and background file scanning does not work.

Note: There is a workaround that allows background file scanning when using **Log-in credentials, save in session**, and that is using Ajax cron mode. (See *Background Jobs.*) Be aware that the Ajax cron mode is triggered by browsing the ownCloud Web GUI.

7.8.3 Public-key Mechanisms

Currently only the RSA mechanism is implemented, where a public/private keypair is generated by ownCloud and the public half shown in the GUI. The keys are generated in the SSH format, and are currently 1024 bits in length. Keys can be regenerated with a button in the GUI.

Authentication	Configuration			
DCA sublished as	Host	Root	Username	ssh-rsa AAAAB3NzaC1:
RSA public key	Generate keys			

7.8.4 OAuth

OAuth 1.0 and OAuth 2.0 are both implemented, but currently limited to the Dropbox and Google Drive backends respectively. These mechanisms require additional configuration at the service provider, where an app ID and app secret are provided and then entered into ownCloud. Then ownCloud can perform an authentication request, establishing the storage connection.

			rt	
٠	sharedropbox	Dropbox		All Users ×
			Access granted	

7.9 Encryption Configuration

The primary purpose of the ownCloud server-side encryption is to protect users' files located on a remote storage, such as Dropbox and Google Drive, and to do it smoothly and seamlessly from within ownCloud.

In ownCloud 9.0 server-side encryption separates the encryption of local and remote storage. Doing so allows you to encrypt files on remote storage, such as Dropbox and Google, without also having to encrypt your home storage on your ownCloud server.

Note: Starting with ownCloud 9.0 we support authenticated encryption for all newly encrypted files. See https://hackerone.com/reports/108082 for more technical information about the impact.

For maximum security make sure to enable "Check for changes: Never." when configuring external storage. With it enabled, ownCloud ignores new files not added via ownCloud, and a malicious external storage administrator could not add new files to the storage without your knowledge. Of course, this is not wise if your external storage is subject to legitimate external changes.

Encryption and decryption are performed on the ownCloud server. ownCloud server-side encryption encrypts files stored on the ownCloud server and files on remote storage that are connected to your ownCloud server. All files stored on remote storages will be decrypted upon retrieval, before serving them to you and anyone you have shared them with.

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

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

ownCloud's server-side encryption generates a strong encryption key, which is unlocked by the user's passwords. Your users don't need to track an extra password, just log in as they normally do. It encrypts only the contents of files, and not filenames and directory structures.

Important: You should regularly backup all encryption keys to prevent permanent data loss.

The encryption keys are stored in the following directories:

Directory	Description
data/ <user>/files_encry</user>	plusers' private keys and all other keys necessary to decrypt the users' files.
data/files_encryption	Private keys and all other keys necessary to decrypt the files stored on a
	system wide external storage.

Note: You can move the keys to a different location. To do so, refer to the Move Key Location section of the documentation.

When encryption is enabled, all files are encrypted and decrypted by the ownCloud application, and stored encrypted on your remote storage. This protects your data on externally hosted storage. The ownCloud admin and the storage admin will see only encrypted files when browsing backend storage.

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

Important: SSL terminates at or before Apache on the ownCloud server. Consequently, all files are in an unencrypted state between the SSL connection termination and the ownCloud code that encrypts and decrypts them. This is, potentially, exploitable by anyone with administrator access to your server. For more information, read: How ownCloud uses encryption to protect your data.

7.9.1 Before Enabling Encryption

Plan very carefully before enabling encryption, because it is not reversible via the ownCloud Web interface. If you lose your encryption keys, your files are *not* recoverable. Always have backups of your encryption keys stored in a safe location, and consider enabling all recovery options. You have more options via the occ command (see *How To Change The Recovery Key Password*)

7.9.2 How To Enable Encryption

The base encryption system is enabled and disabled on your Admin page. First, you must enable this, and then select an encryption module to load. Go to the **Server-side encryption** section of your Admin page and check **Enable encryption**.

After clicking **Enable encryption**, you will see the message "*No encryption module loaded, please load an encryption module in the app menu*". Currently, the only available encryption module is the ownCloud Default Encryption module. So, go to your Apps page to enable the ownCloud Default Encryption module.

Then, return to your Admin page to see that the ownCloud Default Encryption module has been added to the module selector *and* automatically selected. Now you **must** log out and then log back in to initialize your encryption keys.

When you log back in, a checkbox for enabling encryption on your home storage, will now be available — checked by default. Uncheck it to avoid encrypting your home storage.

Enabling Encryption From the Command-line

To enable encryption via the command-line, involves two commands. These are:

```
# Enables the default encryption module app
php occ app:enable encryption
```

Server-side encryption i

Enable server-side encryption

Please read carefully before activating server-side encryption:

- Once encryption is enabled, all files uploaded to the server from that point forward will be encrypted at rest on the server. It will only be possible to disable encryption at a later date if the active encryption module supports that function, and all pre-conditions (e.g. setting a recover key) are met.
- Encryption alone does not guarantee security of the system. Please see ownCloud documentation for more information about how the encryption app works, and the supported use cases.
- Be aware that encryption always increases the file size.
- It is always good to create regular backups of your data, in case of encryption make sure to backup the encryption keys along with your data.

This is the final warning: Do you really want to enable encryption? Enable encryption





Select default encryption module:

Default encryption module

Encryption App is enabled but your keys are not initialized, please log-out and log-in again
Server-side encryption i

Enable server-side encryption

Select default encryption module:

Default encryption module

Encrypt the home storage

Enabling this option encrypts all files stored on the main storage otherwise only files on external storage will be encrypted

Enables encryption
php occ encryption:enable

Note: Please note, the commands have to be run in this order.

7.9.3 Enabling Master Key Based Encryption

To enable master key based encryption:

1. Enable the default encryption module app, using the following command:

php occ app:enable encryption

2. Then enable encryption, using the following command

php occ encryption:enable

3. Then enable the master key, using the following command:

php occ encryption:enable-master-key

Note: When LDAP is configured with a User Home Folder Naming Rule encryption does have some issues. This is a known bug and we are working hard on resolving this issue.

7.9.4 How To Share Encrypted Files

After encryption is enabled, your users must also log out and log back in to generate their personal encryption keys. They will see a yellow warning banner that says "Encryption App is enabled, but your keys are not initialized, please log-out and log-in again."

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

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

7.9.5 How To Encrypt External Mountpoints

You and your users can encrypt individual external mount points. You must have external storage enabled on your Admin page and enabled for your users. Encryption settings can be configured in the mount options for an external storage mount, see *Mount Options (Configuring External Storage (GUI)*)

7.9.6 How To Enable Users File Recovery Keys

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

To avoid all this, make sure you create a recovery key. To do so, go to the Encryption section of your Admin page, where you can set one.



Enable server-side encryption

Select default encryption module:

Default encryption module

Enable recovery key

The recovery key is an extra encryption key that is used to encrypt files. It allows recovery of a user's files if the user forgets his or her password.

•••••	
•••••	Disable recovery key

When you do this, your users will have the option of enabling password recovery on their personal pages. If they do not do this, then the recovery key won't work for them.

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

You may change your recovery key password.

Encryption

Enable password recovery:

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

Enabled

Disabled

File recovery settings updated

Admin Recovery Pas	ssword	Enter the recovery password in order to recover the users		
Group Admin	Quota	files during password change		
Group Admin 🔻	Default	 /var/www/owncloud. 		

Change recovery key password:



7.9.7 How To Change The Recovery Key Password

If you misplace your recovery key password, follow these steps to delete the old recovery share keys and encrypt your files with a new recovery key:

- 1. Delete the recovery key from both data/owncloud_private_keys and "data/public-keys"
- 2. Edit the table oc_appconfig and remove the rows with the config keys: recoveryKeyId and recoveryAdminEnabled for the appid: files_encryption
- 3. Login as admin and activate the recovery key again with a new password. This will generate a new key pair
- 4. All users who used the original recovery key will need to disable it and enable it again

Note: You can only change the recovery key password if you know the original. This is by design, as only admins who know the recovery key password should be able to change it. If not, admins could hijack the recovery key from each other

Warning: Replacing the recovery key will mean that all users will lose the possibility to recover their files until they have applied the new recovery key

7.9.8 Disabling Encryption

To disable encryption, put your ownCloud server into single-user mode, and then disable your encryption module with these commands:

```
occ maintenance:singleuser --on
occ encryption:disable
```

Take it out of single-user mode when you are finished, by using the following command:

```
occ maintenance:singleuser --off
```

Important: You may only disable encryption with by using the occ Encryption Commands. Make sure you have backups of all encryption keys, including those for all your users.

7.9.9 Not All Files Are Encrypted

Only the data in the files in data/user/files are encrypted, not the filenames or folder structures.

In addition, these files are never encrypted:

- Existing files in the trash bin & Versions. Only new and changed files after encryption is enabled are encrypted.
- Image thumbnails from the Gallery app
- Previews from the Files app
- The search index from the full-text search app
- Third-party app data

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

7.9.10 LDAP and Other External User Back-ends

If you use an external user back-end, such as an LDAP or Samba server, and you change a user's password on that back-end, the user will be prompted to change their ownCloud login to match on their next ownCloud login. The user will need both their old and new passwords to do this. If you have enabled the recovery key then you can change a user's password in the ownCloud Users panel to match their back-end password and then — of course — notify the user and give them their new password.

7.9.11 occ Encryption Commands

If you have shell access, you may use the occ command to perform encryption operations, and you have additional options such as decryption and creating a single master encryption key. See *Encryption* for detailed instructions on using occ. Get the current status of encryption and the loaded encryption module:

```
occ encryption:status
  - enabled: false
  - defaultModule: OC_DEFAULT_MODULE
```

This is equivalent to checking Enable server-side encryption on your Admin page:

```
occ encryption:enable
Encryption enabled
```

Default module: OC_DEFAULT_MODULE

List the available encryption modules:

```
occ encryption:list-modules
    - OC_DEFAULT_MODULE: Default encryption module [default*]
```

Select a different default Encryption module (currently the only available module is OC_DEFAULT_MODULE):

```
occ encryption:set-default-module [Module ID].
```

The [module ID] is taken from the encryption:list-modules command. Encrypt all data files for all users. For performance reasons, when you enable encryption on an ownCloud server only new and changed files are encrypted. This command gives you the option to encrypt all files. You must first put your ownCloud server into single-user mode to prevent any user activity until encryption is completed:

occ maintenance:singleuser
Single user mode is currently enabled

Then run occ:

```
occ encryption:encrypt-all
```

You are about to start to encrypt all files stored in your ownCloud. It will depend on the encryption module you use which files get encrypted. Depending on the number and size of your files this can take some time. Please make sure that no users access their files during this process!

Do you really want to continue? (y/n)

When you type y it creates a key pair for each of your users, and then encrypts their files, displaying progress until all user files are encrypted.

Decrypt all user data files, or optionally a single user:

occ encryption:decrypt-all [username]

View current location of keys:

occ encryption:show-key-storage-root
Current key storage root: default storage location (data/)

Move Key Location

Move keys to a different root folder, either locally or on a different server. The folder must already exist, be owned by root and your HTTP group, and be restricted to root and your HTTP group. This example is for Ubuntu Linux. Note that the new folder is relative to your occ directory:

```
mkdir /etc/keys
chown -R root:www-data /etc/keys
chmod -R 0770 /etc/keys
occ encryption:change-key-storage-root ../../../etc/keys
Start to move keys:
    4 [==========]
Key storage root successfully changed to ../../../etc/keys
```

Create a new master key. Use this when you have a single-sign-on infrastructure. Use this only on fresh installations with no existing data, or on systems where encryption has not already been enabled. It is not possible to disable it:

occ encryption:enable-master-key

7.9.12 Encryption migration to ownCloud 8.0

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

7.9.13 Encryption migration to ownCloud 8.1

The encryption backend has changed again in ownCloud 8.1, so you must take some additional steps to migrate encryption correctly. If you do not follow these steps you may not be able to access your files.

Before you start your upgrade, put your ownCloud server into maintenance:singleuser mode (See *Maintenance Mode Configuration*.) You must do this to prevent users and sync clients from accessing files before you have completed your encryption migration.

After your upgrade is complete, follow the steps in *How To Enable Encryption* to enable the new encryption system. Then, click the **Start Migration** button on your Admin page to migrate your encryption keys, or use the occ command. We strongly recommend using the occ command; the **Start Migration** button is for admins who do not have access to the console, for example, installations on shared hosting. This example is for Debian/Ubuntu Linux:

\$ sudo -u www-data php occ encryption:migrate

This example is for Red Hat/CentOS/Fedora Linux:

\$ sudo -u apache php occ encryption:migrate

You must run occ as your HTTP user; see Using the occ Command.

When you are finished, take your ownCloud server out of maintenance:singleuser mode.

7.10 Transactional File Locking

ownCloud's Transactional File Locking mechanism locks files to avoid file corruption during normal operation. It performs these functions:

- Operates at a higher level than the filesystem, so you don't need to use a filesystem that supports locking
- · Locks parent directories so they cannot be renamed during any activity on files inside the directories
- Releases locks after file transactions are interrupted, for example when a sync client loses the connection during an upload
- Manages locking and releasing locks correctly on shared files during changes from multiple users
- · Manages locks correctly on external storage mounts
- · Manages encrypted files correctly

What Transactional File locking is not for: it is not for preventing collisions in collaborative document editing (see *Configuring the Collaborative Documents App* to learn about collaboration with the Documents app), nor will it prevent multiple users from editing the same document, or give notice that other users are working on the same document. Multiple users can open and edit a file at the same time and Transactional File locking does not prevent this. Rather, it prevents simultaneous file saving.

Note: Transactional file locking is in ownCloud core, and replaces the old File Locking app. The File Locking app has been removed from ownCloud in version 8.2.1. If your ownCloud server still has the File Locking app, you must visit your Apps page to verify that it is disabled; the File Locking app and Transactional File Locking cannot both operate at the same time.

File locking is enabled by default, using the database locking backend. This places a significant load on your database. Using memcache.locking relieves the database load and improves performance. Admins of ownCloud servers with heavy workloads should install a memcache. (See *Configuring Memory Caching*.)

To use a memcache with Transactional File Locking, you must install the Redis server and corresponding PHP module. After installing Redis you must enter a configuration in your config.php file like this example:

```
'filelocking.enabled' => true,
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => array(
    'host' => 'localhost',
    'port' => 6379,
    'timeout' => 0.0,
    'password' => '', // Optional, if not defined no password will be used.
    ),
```

Note: For enhanced security it is recommended to configure Redis to require a password. See http://redis.io/topics/security for more information.

If you want to configure Redis to listen on an Unix socket (which is recommended if Redis is running on the same system as ownCloud) use this example config.php configuration:

See config.sample.php to see configuration examples for Redis, and for all supported memcaches.

If you are on Ubuntu you can follow this guide for a complete installation from scratch.

Learn more about Reds at Redis. Memcached, the popular distributed memory caching system, is not suitable for the new file locking because it is not designed to store locks, and data can disappear from the cache at any time. Redis is a key-value store, and it guarantees that cached objects are available for as long as they are needed.

Debian Jesse users, please see this Github discussion if you have problems with LDAP authentication.

7.11 Previews Configuration

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

The following image shows some examples of previews of various file types.

#	Pho	ptos +
		Name 🔺
	Openhiller Onere Hann Ger Hanner ger Hanner ger	client-7.png
*	Þ	external storage in ownCloud 8.1.mp4
	Л	installation-of-owncloud-8-1.mp3
	17. S	Paris.jpg
	×	San Francisco.jpg
	A. M.	Squirrel.jpg
		whats_new.rst

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

- Images files
- Cover of MP3 files
- Text documents

Note: Older versions of ownCloud also supported the preview generation of other file types such as PDF, SVG or various office documents. Due to security concerns those providers have been disabled by default and are considered unsupported. While those providers are still available, we discourage enabling them, and they are not documented.

7.11.1 Parameters

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

Disabling previews:

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

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

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

Maximum preview size:

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

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

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

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

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

Maximum scale factor:

If a lot of small pictures are stored on the ownCloud instance and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, pictures are upscaled to 10 times the original size:

<?php
'preview_max_scale_factor' => 10,

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

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

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

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

7.12 Controlling File Versions and Aging

The Versions app (files_versions) expires old file versions automatically to ensure that users don't exceed their storage quotas. This is the default pattern used to delete old versions:

- For the first second we keep one version
- For the first 10 seconds ownCloud keeps one version every 2 seconds
- · For the first minute ownCloud keeps one version every 10 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 is created.

The Versions app never uses more that 50% of the user's currently available free space. If the stored versions exceed this limit, ownCloud deletes the oldest file versions until it meets the disk space limit again.

You may alter the default pattern in config.php. The default setting is auto, which sets the default pattern:

```
'versions_retention_obligation' => 'auto',
```

Additional options are:

- D, auto Keep versions at least for D days, apply expiration rules to all versions that are older than D days
- auto, D Delete all versions that are older than D days automatically, delete other versions according to expiration rules
- D1, D2 Keep versions for at least D1 days and delete when they exceed D2 days.
- disabled Disable Versions; no files will be deleted.

7.12.1 Enterprise File Retention

Enterprise customers have additional tools for managing file retention policies; see Advanced File Tagging With the Workflow App (Enterprise only).

CHAPTER

EIGHT

DATABASE CONFIGURATION

8.1 Converting Database Type

You can convert a SQLite database to a more performing MySQL, MariaDB or PostgreSQL database with the own-Cloud command line tool. SQLite is good for testing and simple single-user ownCloud servers, but it does not scale for multiple-user production users.

Note: ownCloud Enterprise edition does not support SQLite.

8.1.1 Run the conversion

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

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

The Options

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

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

For example

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

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

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

8.1.2 Unconvertible Tables

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

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

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

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

8.2 Database Configuration

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

- MySQL / MariaDB
- PostgreSQL
- Oracle (ownCloud Enterprise edition only)

The MySQL or MariaDB databases are the recommended database engines.

8.2.1 Requirements

Choosing to use MySQL / MariaDB, PostgreSQL, or Oracle (ownCloud Enterprise edition only) as your database requires that you install and set up the server software first. (Oracle users, see *Oracle Database Setup*.)

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

MySQL / MariaDB with Binary Logging Enabled

ownCloud is currently using a TRANSACTION_READ_COMMITTED transaction isolation to avoid data loss under high load scenarios (e.g. by using the sync client with many clients/users and many parallel operations). This requires

a disabled or correctly configured binary logging when using MySQL or MariaDB. Your system is affected if you see the following in your log file during the installation or update of ownCloud:

An unhandled exception has been thrown: exception 'PDOException' with message 'SQL-STATE[HY000]: General error: 1665 Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging. InnoDB is limited to row-logging when transaction isolation level is READ COMMITTED or READ UNCOMMITTED.'

There are two solutions. One is to disable binary logging. Binary logging records all changes to your database, and how long each change took. The purpose of binary logging is to enable replication and to support backup operations.

The other is to change the BINLOG_FORMAT = STATEMENT in your database configuration file, or possibly in your database startup script, to BINLOG_FORMAT = MIXED. See Overview of the Binary Log and The Binary Log for detailed information.

MySQL / MariaDB "READ COMMITED" transaction isolation level

As discussed above ownCloud is using the TRANSACTION_READ_COMMITTED transaction isolation level. Some database configurations are enforcing other transaction isolation levels. To avoid data loss under high load scenarios (e.g. by using the sync client with many clients/users and many parallel operations) you need to configure the transaction isolation level accordingly. Please refer to the MySQL manual for detailed information.

MySQL / MariaDB storage engine

Since ownCloud 7 only InnoDB is supported as a storage engine. There are some shared hosters that do not support InnoDB and only MyISAM. Running ownCloud on such an environment is not supported.

8.2.2 Parameters

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

Configuring a MySQL or MariaDB Database

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

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

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

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

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

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

You can quit the prompt by entering:

quit

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

<?php

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

PostgreSQL Database

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

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so
[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```

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

sudo -u postgres psql -d template1

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

CREATE USER username CREATEDB; CREATE DATABASE owncloud OWNER username;

You can quit the prompt by entering:

/d

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

<?php

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

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

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

To start the postgres command line mode use:

psql -hlocalhost -Upostgres

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

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

You can quit the prompt by entering:

/d

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

<?php

```
"dbtype" => "pgsql",
"dbname" => "owncloud",
```

```
"dbuser" => "username",
"dbpassword" => "password",
"dbhost" => "localhost",
"dbtableprefix" => "oc_",
```

8.2.3 Troubleshooting

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

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

ping db.server.dom

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

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

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

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

MySQL:

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

mysql -uUSERNAME -p

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

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

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

PostgreSQL:

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

psql -Uusername -downcloud

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

```
psql -Uusername -downcloud -h HOSTNAME
```

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

Useful SQL commands

Show Database Users:

MySQL : SELECT User,Host FROM mysql.user; PostgreSQL: SELECT * FROM pg_user;

Show available Databases:

MySQL : SHOW DATABASES; PostgreSQL: \l

Show ownCloud Tables in Database:

MySQL : USE owncloud; SHOW TABLES; PostgreSQL: \c owncloud; \d

Quit Database:

MySQL : quit PostgreSQL: \q

CHAPTER

MIMETYPES MANAGEMENT

9.1 Mimetype Aliases

ownCloud allows you to create aliases for mimetypes, so that you can display custom icons for files. For example, you might want a nice audio icon for audio files instead of the default file icon.

By default ownCloud is distributed with owncloud/resources/config/mimetypealiases.dist.json. Do not modify this file, as it will be replaced when ownCloud is updated. Instead, create your own owncloud/config/mimetypealiases.json file with your custom aliases. Use the same syntax as in owncloud/resources/config/mimetypealiases.dist.json.

Once you have made changes to your mimetypealiases.json, use the occ command to propagate the changes through the system. This example is for Ubuntu Linux:

\$ sudo -u www-data php occ maintenance:mimetype:update-js

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

Some common mimetypes that may be useful in creating aliases are:

image Generic image

image/vector Vector image

audio Generic audio file

x-office/document Word processed document

x-office/spreadsheet Spreadsheet

x-office/presentation Presentation

text Generic text document

text/code Source code

9.2 Mimetype mapping

ownCloud allows administrators to specify the mapping of a file extension to a mimetype. For example files ending in mp3 map to audio/mpeg. Which then in turn allows ownCloud to show the audio icon.

By default ownCloud comes with mimetypemapping.dist.json. This is a simple json array. Administrators should not update this file as it will get replaced on upgrades of ownCloud. Instead the file mimetypemapping.json should be created and modified, this file has precedence over the shipped file.

9.3 Icon retrieval

When an icon is retrieved for a mimetype, if the full mimetype cannot be found, the search will fallback to looking for the part before the slash. Given a file with the mimetype 'image/my-custom-image', if no icon exists for the full mimetype, the icon for 'image' will be used instead. This allows specialised mimetypes to fallback to generic icons when the relevant icons are unavailable.

CHAPTER

MAINTENANCE

10.1 Maintenance Mode Configuration

You must put your ownCloud server into maintenance mode before performing upgrades, and for performing troubleshooting and maintenance. Please see *Using the occ Command* to learn how to put your server into the various maintenance modes (maintenance:mode, maintenance:singleuser, and maintenance:repair) with the occ command.

maintenance:mode locks the sessions of logged-in users and prevents new logins. This is the mode to use for upgrades. You must run occ as the HTTP user, like this example on Ubuntu Linux:

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

You may also put your server into this mode by editing config/config.php. Change "maintenance" => false to "maintenance" => true:

<?php

"maintenance" => true,

Then change it back to false when you are finished.

10.2 Backing up ownCloud

When you backup your ownCloud server, there are four things that you need to copy:

- 1. Your config/ directory.
- 2. Your data/ directory.
- 3. Your ownCloud database.
- 4. Your custom theme files, if you have any. (See Theming ownCloud)

When you install your ownCloud server from our Open Build Service) packages (or from distro packages, which we do not recommend) **do not backup your ownCloud server files**, which are the other files in your owncloud/ directory such as core/, 3rdparty/, apps/, assets/, lib/, and all the rest of the ownCloud files. If you restore these files from backup they may not be in sync with the current package versions, and will fail the code integrity check. This may also cause other errors, such as white pages.

When you install ownCloud from the source tarballs this will not be an issue, and you can safely backup your entire ownCloud installation, with the exception of your ownCloud database. Databases cannot be copied, but you must use the database tools to make a correct database dump.

To restore your ownCloud installation from backup, see Restoring ownCloud .

10.2.1 Backing Up the config/ and data/ Directories

Simply copy your config/ and data/ folder to a place outside of your ownCloud environment. This example uses rsync to copy the two directories to /backupdir:

rsync -Aax config data /oc-backupdir/

There are many ways to backup normal files, and you may use whatever method you are accustomed to.

10.2.2 Backup Database

You can't just copy a database, but must use the database tools to make a correct database dump.

MySQL/MariaDB

MySQL or MariaDB, which is a drop-in MySQL replacement, is the recommended database engine. To backup MySQL/MariaDB:

```
mysqldump --single-transaction -h [server] -u [username] -p[password] [db_name] > owncloud-dbbackup_
```

Example:

```
mysqldump --single-transaction -h localhost -u username -ppassword owncloud > owncloud-dbbackup_'date
```

SQLite

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

PostgreSQL

PGPASSWORD="password" pg_dump [db_name] -h [server] -U [username] -f owncloud-dbbackup_'date +"%Y%m%

10.2.3 Restoring Files From Backup When Encryption Is Enabled

If you need to restore files from backup, which were backed up when encryption was enabled, here's how to do it.

Note: This is effective from at least version v8.2.7 of ownCloud onwards. Also, this is **not officially supported**. ownCloud officially supports either restoring the full backup or restoring nothing — not restoring individual parts of it.

- 1. Restore the file from backup
- 2. Restore the file's encryption keys from backup
- 3. Run occ files:scan; this makes the scanner find it. Note that, in the DB it will (1) have the "size" set to the encrypted size, which is wrong (and bigger) and (2) the "encrypted" flag will be set to 0
- 4. Update the "encrypted" flag to 1 in the DB to all *files* under files/path, but **not** directories. Setting the flag to 1 tells the encryption application that the file is encrypted and needs to be processed.

Note: There's no need to update the encrypted flag for files in either "files_versions" or "files_trashbin", because these aren't scanned or found by occ files:scan.

5. Download the file once as the user; the file's size will be corrected automatically

This process might not be suitable across all environments. If it's not suitable for yours, you might need to run an OCC command that does the scanning. But, that will require the user's password or recovery key.

10.3 How to Upgrade Your ownCloud Server

There are three ways to upgrade your ownCloud server:

- Using your Linux package manager with our official ownCloud repositories. This is the recommended method.
- With the *Updater App* (Server Edition only). Recommended for shared hosters, and for users who want an easy way to track different release channels. (It is not available and not supported on the Enterprise edition.)
- Manually upgrading with the ownCloud .tar archive from owncloud.org/install/.
- Manually upgrading is also an option for users on shared hosting; download and unpack the ownCloud tarball to your PC. Delete your existing ownCloud files, except data/ and config/ files, on your hosting account. Then transfer the new ownCloud files to your hosting account, again preserving your existing data/ and config/ files.
- Enterprise customers will use their Enterprise software repositories to maintain their ownCloud servers, rather than the Open Build Service. Please see *Installing & Upgrading ownCloud Enterprise Edition* for more information.

Warning: When upgrading from oC 9.0 to 9.1 with existing Calendars or Adressbooks please have a look at the *ownCloud 9.0 Release Notes* of oC 9.0 for important info about the needed migration steps during that upgrade.

When an update is available for your ownCloud server, you will see a notification at the top of your ownCloud Web interface. When you click the notification it brings you here, to this page.

It is best to keep your ownCloud server upgraded regularly, and to install all point releases and major releases without skipping any of them, as skipping releases increases the risk of errors. Major releases are 8.0, 8.1, 8.2, and 9.0. Point releases are intermediate releases for each major release. For example, 8.0.9 and 8.1.3 are point releases. Skipping major releases is not supported.

Upgrading is disruptive. Your ownCloud server will be put into maintenance mode, so your users will be locked out until the upgrade is completed. Large installations may take several hours to complete the upgrade.

Warning: Downgrading is not supported and risks corrupting your data! If you want to revert to an older ownCloud version, make a new, fresh installation and then restore your data from backup. Before doing this, file a support ticket (if you have paid support) or ask for help in the ownCloud forums to see if your issue can be resolved without downgrading.

10.3.1 Prerequisites

You should always maintain *regular backups* and make a fresh backup before every upgrade.

Then review third-party apps, if you have any, for compatibility with the new ownCloud release. Any apps that are not developed by ownCloud show a 3rd party designation. **Install unsupported apps at your own risk**. Then, before the upgrade, all 3rd party apps must be disabled. After the upgrade is complete you may re-enable them.

10.3.2 Previous ownCloud Releases

You'll find previous ownCloud releases in the ownCloud Server Changelog.

10.3.3 Reverse Upgrade

If you need to reverse your upgrade, see Restoring ownCloud.

10.3.4 Troubleshooting

When upgrading ownCloud and you are running MySQL or MariaDB with binary logging enabled, your upgrade may fail with these errors in your MySQL/MariaDB log:

```
An unhandled exception has been thrown:
exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665
Cannot execute statement: impossible to write to binary log since
BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited
to row-based logging. InnoDB is limited to row-logging when transaction
isolation level is READ COMMITTED or READ UNCOMMITTED.'
```

Please refer to MySQL / MariaDB with Binary Logging Enabled on how to correctly configure your environment.

Occasionally, files do not show up after a upgrade. A rescan of the files can help:

sudo -u www-data php console.php files:scan --all

See the owncloud.org support page for further resources for both home and enterprise users.

Sometimes, ownCloud can get *stuck in a upgrade*. This is usually due to the process taking too long and encountering a PHP time-out. Stop the upgrade process this way:

sudo -u www-data php occ maintenance:mode --off

Then start the manual process:

sudo -u www-data php occ upgrade

If this does not work properly, try the repair function:

sudo -u www-data php occ maintenance:repair

10.3.5 Migration Test

Before completing the upgrade, ownCloud first runs a simulation by copying all database tables to new tables, and then performs the upgrade on them, to ensure that the upgrade will complete correctly. The copied tables are deleted after the upgrade. This takes twice as much time, which on large installations can be many hours, so you can omit this step (by risking a failed upgrade) with the --skip-migration-test option, like this example on CentOS:

\$ sudo -u apache php occ upgrade --skip-migration-test

or this example on Ubuntu:

\$ sudo -u www-data php occ upgrade --skip-migration-test

See Using the occ Command to learn more.

10.3.6 Encryption migration from oC 7.0 to 8.0 and 8.0 to 8.1

The encryption backend was changed twice between ownCloud 7.0 and 8.0 as well as between 8.0 and 8.1. If you're upgrading from these older versions please refer to *Encryption migration to ownCloud 8.0* for the needed migration steps.

10.3.7 Debian Migration to Official ownCloud Packages

As of March 2016 Debian will not include ownCloud packages. Debian users can migrate to the official ownCloud packages by following this guide, Upgrading ownCloud on Debian Stable to official packages.

10.4 Upgrade ownCloud From Packages

10.4.1 Upgrade Quickstart

The best method for keeping ownCloud current on Linux servers is by configuring your system to use ownCloud's Open Build Service repository. Then stay current by using your Linux package manager to install fresh ownCloud packages. After installing upgraded packages you must run a few more steps to complete the upgrade. These are the basic steps to upgrading ownCloud:

Warning: Make sure that you don't skip a major release when upgrading via repositories. For example you can't upgrade from 8.1.x to 9.0.x directly as you would skip the 8.2.x major release. See *Upgrading Across Skipped Releases* for more information.

- *Disable* all third-party apps.
- Make a fresh backup.
- Upgrade your ownCloud packages.
- Run occ upgrade (Optionally disable the Migration Test which might take a long time on large installations).
- Apply strong permissions to your ownCloud directories.
- Take your ownCloud server out of maintenance mode.
- Re-enable third-party apps.

Warning: When upgrading from oC 9.0 to 9.1 with existing Calendars or Addressbooks please have a look at the *9.0 release notes* for important information about the needed migration steps during that upgrade.

10.4.2 Upgrade Tips

Upgrading ownCloud from our Open Build Service repository is just like any normal Linux upgrade. For example, on Debian or Ubuntu Linux this is the standard system upgrade command:

apt-get update && apt-get upgrade

Or you can upgrade just ownCloud with this command:

apt-get update && apt-get install owncloud

On Fedora, CentOS, and Red Hat Linux use yum to see all available updates:

yum check-update

You can apply all available updates with this command:

yum update

Or update only ownCloud:

yum update owncloud

Your Linux package manager only downloads the current ownCloud packages. Then your ownCloud server is immediately put into maintenance mode. You may not see this until you refresh your ownCloud page.



Then use occ to complete the upgrade. You must run occ as your HTTP user. This example is for Debian/Ubuntu:

sudo -u www-data php occ upgrade

This example is for CentOS/RHEL/Fedora:

sudo -u apache php occ upgrade

Optionally disable the Migration Test which might take a long time on large installations.

See Using the occ Command to learn more.

10.4.3 Setting Strong Directory Permissions

After upgrading, verify that your ownCloud directory permissions are set according to *Setting Strong Directory Permissions*.

10.4.4 Upgrading Across Skipped Releases

It is best to update your ownCloud installation with every new point release (e.g. 8.1.10), and to never skip any major release (e.g. don't skip 8.2.x between 8.1.x and 9.0.x). If you have skipped any major release you can bring your ownCloud current with these steps:

- 1. Add the repository of your current version (e.g. 8.1.x)
- 2. Upgrade your current version to the latest point release (e.g. 8.1.10) via your package manager
- 3. Run the occ upgrade routine (see Upgrade Quickstart above)
- 4. Add the repository of the next major release (e.g. 8.2.x)
- 5. Upgrade your current version to the next major release (e.g. 8.2.8) via your package manager
- 6. Run the occ upgrade routine (see Upgrade Quickstart above)
- 7. Repeat from step 4 until you reach the last available major release (e.g. 9.1.x)

You'll find repositories of previous ownCloud major releases in the ownCloud Server Changelog.

10.5 Upgrading ownCloud with the Updater App

The Updater app automates many of the steps of upgrading an ownCloud installation. It is useful for installations that do not have root access, such as shared hosting, for installations with a smaller number of users and data, and it automates updating *manual installations*.

Warning: When upgrading from oC 9.0 to 9.1 with existing Calendars or Adressbooks please have a look at the *ownCloud 9.0 Release Notes* of oC 9.0 for important info about the needed migration steps during that upgrade.

New in 9.0, the Updater app has command-line options.

Note: The Updater app is not enabled and not supported in ownCloud Enterprise edition.

The Updater app is **not included** in the Linux packages on our Open Build Service, but only in the tar and zip archives. When you install ownCloud from packages you should keep it updated with your package manager.

Downgrading is not supported and risks corrupting your data! If you want to revert to an older ownCloud version, install it from scratch and then restore your data from backup. Before doing this, file a support ticket (if you have paid support) or ask for help in the ownCloud forums to see if your issue can be resolved without downgrading.

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

The Updater app performs these operations:

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

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

- 1. You should see a notification at the top of any ownCloud page when there is a new update available.
- 2. Even though the Updater app backs up important directories, you should always have your own current backups (See *Backing up ownCloud* for details.)
- 3. Verify that the HTTP user on your system can write to your whole ownCloud directory; see the *Setting Permissions for Updating* section below.
- 4. Navigate to your Admin page and click the **Update Center** button under Updater. This takes you to the Updater control panel.
- 5. Click Update, and carefully read the messages. If there are any problems it will tell you. The most common issue is directory permissions; your HTTP user needs write permissions to your whole ownCloud directory. (See *Setting Strong Directory Permissions*.) Another common issue is SELinux rules (see *SELinux Configuration*.) Otherwise you will see messages about checking your installation and making backups.
- 6. Click Proceed, and then it performs the remaining steps, which takes a few minutes.
- 7. If your directory permissions are correct, a backup was made, and downloading the new ownCloud archive succeeded you will see the following screen. Click the Start Update button to complete your update:



Note: If you have a large ownCloud installation and have shell access, you should use the occ upgrade command, running it as your HTTP user, instead of clicking the Start Update button, in order to avoid PHP timeouts.

This example is for Ubuntu Linux:

\$ sudo -u www-data php occ upgrade

Optionally disable the *Migration Test* which might take a long time on large installations.

See Using the occ Command to learn more.

8. It runs for a few minutes, and when it is finished displays a success message, which disappears after a short time.

Refresh your Admin page to verify your new version number. In the Updater section of your Admin page you can see the current status and backups. These are backups of your old and new ownCloud installations, and do not contain your data files. If your update works and there are no problems you can delete the backups from this screen.

If the update fails, then you must update manually. (See Manually upgrading.)

10.5.1 Setting Permissions for Updating

For hardened security we highly recommend setting the permissions on your ownCloud directory as strictly as possible. These commands should be executed immediately after the initial installation. Please follow the steps in *Setting Strong Directory Permissions*.

These strict permissions will prevent the Updater app from working, as it needs your whole ownCloud directory to be owned by the HTTP user. Run this script to set the appropriate permissions for updating. Replace the ocpath variable with the path to your ownCloud directory, and replace the htuser and htgroup variables with your HTTP user and group.:

```
#!/bin/bash
# Sets permissions of the owncloud instance for updating
ocpath='/var/www/owncloud'
htuser='www-data'
htgroup='www-data'
```

```
chown -R ${htuser}:${htgroup} ${ocpath}
```

You can find your HTTP user in your HTTP server configuration files. Or you can use *PHP Version and Information* (Look for the **User/Group** line).

- The HTTP user and group in Debian/Ubuntu is www-data.
- The HTTP user and group in Fedora/CentOS is apache.
- The HTTP user and group in Arch Linux is http.
- The HTTP user in openSUSE is www.run, and the HTTP group is www.

After the update is completed, re-apply the strong directory permissions immediately by running the script in *Setting Strong Directory Permissions*.

10.5.2 Command Line Options

The Updater app includes command-line options to automate updates, to create checkpoints and to roll back to older checkpoints. You must run it as your HTTP user. This example on Ubuntu Linux displays command options:

sudo -u www-data php updater/application.php list

See usage for commands, like this example for the upgrade: checkpoint command:

sudo -u www-data php updater/application.php upgrade:checkpoint -h

You can display a help summary:

sudo -u www-data php updater/application.php --help

When you run it without options it runs a system check:

sudo -u www-data php owncloud/updater/application.php ownCloud updater 1.0 - CLI based ownCloud server upgrades Checking system health. - file permissions are ok. Current version is 9.0.0.12 No updates found online. Done

Create a checkpoint:

sudo -u www-data php updater/application.php upgrade:checkpoint --create Created checkpoint 9.0.0.12-56d5e4e004964

List checkpoints:

sudo -u www-data php updater/application.php upgrade:checkpoint --list

Restore an earlier checkpoint:

Add a line like this to your crontab to automatically create daily checkpoints:

```
2 15 * * * sudo -u www-data php /path/to/owncloud/updater/application.php upgrade:checkpoint --create > /dev/null 2>&1
```

10.6 Manual ownCloud Upgrade

Warning: When upgrading from oC 9.0 to 9.1 with existing Calendars or Adressbooks please have a look at the *ownCloud 9.0 Release Notes* of oC 9.0 for important info about the needed migration steps during that upgrade.

Always start by making a fresh backup and disabling all 3rd party apps.

Put your server in maintenance mode. This prevents new logins, locks the sessions of logged-in users, and displays a status screen so users know what is happening. There are two ways to do this, and the preferred method is to use the *occ command*, which you must run as your HTTP user. This example is for Ubuntu Linux:

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

The other way is by entering your config.php file and changing 'maintenance' => false, to 'maintenance' => true,.

- 1. Back up your existing ownCloud Server database, data directory, and config.php file. (See *Backing up ownCloud*.)
- Download and unpack the latest ownCloud Server release (Archive file) from owncloud.org/install/ into an empty directory outside of your current installation.

Note: To unpack your new tarball, run: tar xjf owncloud-[version].tar.bz2

Note: Enterprise users must download their new ownCloud archives from their accounts on https://customer.owncloud.com/owncloud/

- 3. Stop your Web server.
- 4. Rename your current ownCloud directory, for example owncloud-old.
- 5. Unpacking the new archive creates a new owncloud directory populated with your new server files. Copy this directory and its contents to the original location of your old server, for example /var/www/, so that once again you have /var/www/owncloud.
- 6. Copy the config.php file from your old ownCloud directory to your new ownCloud directory.

- 7. If you keep your data/ directory in your owncloud/ directory, copy it from your old version of ownCloud to your new owncloud/. If you keep it outside of owncloud/ then you don't have to do anything with it, because its location is configured in your original config.php, and none of the upgrade steps touch it.
- 8. If you are using 3rd party applications, look in your new owncloud/apps/ directory to see if they are there. If not, copy them from your old apps/ directory to your new one. Make sure the directory permissions of your third party application directories are the same as for the other ones.
- 9. Restart your Web server.
- 10. Now launch the upgrade from the command line using occ, like this example on CentOS Linux:

sudo -u apache php occ upgrade

Optionally disable the Migration Test which might take a long time on large installations.

See Using the occ Command to learn more.

11. The upgrade operation takes a few minutes to a few hours, depending on the size of your installation. When it is finished you will see a success message, or an error message that will tell where it went wrong.

Assuming your upgrade succeeded, disable the maintenance mode:

sudo -u www-data php occ maintenance:mode --off

Login and take a look at the bottom of your Admin page to verify the version number. Check your other settings to make sure they're correct. Go to the Apps page and review the core apps to make sure the right ones are enabled. Reenable your third-party apps. Then apply strong permissions to your ownCloud directories (*Setting Strong Directory Permissions*).

10.7 Restoring ownCloud

When you install ownCloud from packages, follow these steps to restore your ownCloud installation. Start with a fresh ownCloud package installation in a new, empty directory. Then restore these items from your backup (see *Backing up ownCloud*):

- 1. Your config/ directory.
- 2. Your data/ directory.
- 3. Your ownCloud database.
- 4. Your custom theme files, if you have any. (See Theming ownCloud)

When you install ownCloud from the source tarballs you may safely restore your entire ownCloud installation from backup, with the exception of your ownCloud database. Databases cannot be copied, but you must use the database tools to make a correct restoration.

When you have completed your restoration, see Setting Strong Permissions.

10.7.1 Restore Directories

Simply copy your configuration and data folder to your ownCloud environment. You could use this command, which restores the backup example in *Backing up ownCloud*:

rsync -Aax config data /var/www/owncloud/

There are many ways to restore normal files from backups, and you may use whatever method you are accustomed to.

10.7.2 Restore Database

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

MySQL

MySQL is the recommended database engine. To restore MySQL:

mysql -h [server] -u [username] -p[password] [db_name] < owncloud-dbbackup.bak</pre>

SQLite

```
rm data/owncloud.db
sqlite3 data/owncloud.db < owncloud-dbbackup.bak</pre>
```

PostgreSQL

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

10.8 Migrating to a Different Server

If the need arises ownCloud can be migrated to a different server. A typical use case would be a hardware change or a migration from the virtual appliance to a physical server. All migrations have to be performed with ownCloud offline and no accesses being made. Online migration is supported by ownCloud only when implementing industry-standard clustering and high-availability solutions before ownCloud is installed for the first time.

To start, let us be specific about the use case. A configured ownCloud instance runs reliably on one machine. For some reason (e.g. more powerful machine is available, but a move to a clustered environment not yet needed) the instance needs to be moved to a new machine. Depending on the size of the ownCloud instance the migration might take several hours. As a prerequisite it is assumed that the end users reach the ownCloud instance via a virtual hostname (a CNAME record in DNS) which can be pointed at the new location. It is also assumed that the authentication method (e.g. LDAP) remains the same after the migration.

Warning: At NO TIME any changes to the **ORIGINAL** system are required **EXCEPT** putting ownCloud into maintenance mode.

This ensures, should anything unforeseen happen, you can go back to your existing installation and provide your users with a running ownCloud while debugging the problem.

- Set up the new machine with your desired Linux distribution. At this point you can either install ownCloud
 manually via the compressed archive (see *Manual Installation on Linux*, or with your Linux package manager
 (see *Preferred Linux Installation Method*).
- On the original machine turn on maintenance mode and then stop ownCloud. After waiting for 6-7 minutes for all sync clients to register the server as in maintenance mode, stop the application and/or Web server that serves ownCloud. (See *Maintenance Commands*.)
- 3. Create a dump from the database and copy it to the new machine, and import it into the new database (See *Backing up ownCloud* and *Restoring ownCloud*).

4. Copy ONLY your data, configuration and database files from your original ownCloud instance to the new machine (See *Backing up ownCloud* and *Restoring ownCloud*).

Note: You must keep the data/ directory's original filepath. Do not change this!

- 5. The data files should keep their original timestamp (can be done by using rsync with -t option) otherwise the clients will re-download all the files after the migration. This step might take several hours, depending on your installation.
- 6. With ownCloud still in maintenance mode (confirm!) and **BEFORE** changing the CNAME record in the DNS start up the database, Web server / application server on the new machine and point your Web browser to the migrated ownCloud instance. Confirm that you see the maintenance mode notice, that a logfile entry is written by both the Web server and ownCloud and that no error messages occur. Then take ownCloud out of maintenance mode and repeat. Log in as admin and confirm normal function of ownCloud.

7. Change the CNAME entry in the DNS to point your users to the new location.

CHAPTER

ELEVEN

ISSUES AND TROUBLESHOOTING

11.1 General Troubleshooting

If you have trouble installing, configuring or maintaining ownCloud, please refer to our community support channels:

• The ownCloud Forums

Note: The ownCloud forums have a FAQ category where each topic corresponds to typical mistakes or frequently occurring issues

- The ownCloud User mailing list
- The ownCloud IRC chat channel irc://#owncloud@freenode.net on freenode.net, also accessible via webchat

Please understand that all these channels essentially consist of users like you helping each other out. Consider helping others out where you can, to contribute back for the help you get. This is the only way to keep a community like ownCloud healthy and sustainable!

If you are using ownCloud in a business or otherwise large scale deployment, note that ownCloud Inc. offers the Enterprise Edition with commercial support options.

11.1.1 Bugs

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

- Search for a solution (see the options above)
- Double-check your configuration

If you can't find a solution, please use our bugtracker. You can generate a configuration report with the *occ config command*, with passwords automatically obscured.

11.1.2 General Troubleshooting

Check the ownCloud System Requirements, especially supported browser versions.

When you see warnings about code integrity, refer to Code Signing.

Disable 3rdparty / non-shipped apps

It might be possible that 3rd party / non-shipped apps are causing various different issues. Always disable 3rd party apps before upgrades, and for troubleshooting. Please refer to the *Apps Commands* on how to disable an app from command line.

ownCloud Logfiles

In a standard ownCloud installation the log level is set to Normal. To find any issues you need to raise the log level to All in your config.php file, or to **Everything** on your ownCloud Admin page. Please see *Logging Configuration* for more information on these log levels.

Some logging - for example JavaScript console logging - needs debugging enabled. Edit config/config.php and change 'debug' => false, to 'debug' => true, Be sure to change it back when you are finished.

For JavaScript issues you will also need to view the javascript console. All major browsers have developer tools for viewing the console, and you usually access them by pressing F12. For Firefox we recommend to installing the Firebug extension.

Note: The logfile of ownCloud is located in the data directory owncloud/data/owncloud.log.

PHP Version and Information

You will need to know your PHP version and configurations. To do this, create a plain-text file named **phpinfo.php** and place it in your Web root, for example /var/www/html/phpinfo.php. (Your Web root may be in a different location; your Linux distribution documentation will tell you where.) This file contains just this line:

<?php phpinfo(); ?>

Open this file in a Web browser by pointing your browser to localhost/phpinfo.php:

localhost/phpinf	o.php 🗸 C 🔍 Search		☆自	Ŧ	⋒	»	≡
PHP Versio	n 5.5.9-1ubuntu4.9		P	h			Ĉ
System	Linux studio 3.13.0-37-generic #64-Ubuntu SM 2014 x86_64	1P Mon	Sep 22 21	:28:38	UTC		
Build Date	Арг 17 2015 11:41:17						- 1
Server API	Apache 2.0 Handler					1	

Your PHP version is at the top, and the rest of the page contains abundant system information such as active modules, active .ini files, and much more. When you are finished reviewing your information you must delete phpinfo.php, or move it outside of your Web directory, because it is a security risk to expose such sensitive data.

Debugging Sync Issues

Warning: The data directory on the server is exclusive to ownCloud and must not be modified manually.
Disregarding this can lead to unwanted behaviours like:

- Problems with sync clients
- Undetected changes due to caching in the database

If you need to directly upload files from the same server please use a WebDAV command line client like cadaver to upload files to the WebDAV interface at:

https://example.com/owncloud/remote.php/dav

Common problems / error messages

Some common problems / error messages found in your logfiles as described above:

- SQLSTATE[HY000] [1040] Too many connections -> You need to increase the connection limit of your database, please refer to the manual of your database for more information.
- SQLSTATE[HY000]: General error: 5 database is locked -> You're using SQLite which can't handle a lot of parallel requests. Please consider converting to another database like described in *Converting Database Type*.
- SQLSTATE[HY000]: General error: 2006 MySQL server has gone away -> The database request takes too long and therefore the MySQL server times out. Its also possible that the server is dropping a packet that is too large. Please refer to the manual of your database for how to raise the config options wait_timeout and/or max_allowed_packet.
- SQLSTATE[HY000] [2002] No such file or directory -> There is a problem accessing your SQLite database file in your data directory (data/owncloud.db). Please check the permissions of this folder/file or if it exists at all. If you're using MySQL please start your database.
- Connection closed / Operation cancelled -> This could be caused by wrong KeepAlive settings within your Apache config. Make sure that KeepAlive is set to On and also try to raise the limits of KeepAliveTimeout and MaxKeepAliveRequests.
- No basic authentication headers were found -> This error is shown in your data/owncloud.log file. Some Apache modules like mod_fastcgi, mod_fcgid or mod_proxy_fcgi are not passing the needed authentication headers to PHP and so the login to own-Cloud via WebDAV, CalDAV and CardDAV clients is failing. Information on how to correctly configure your environment can be found at the forums.

11.1.3 Troubleshooting Web server and PHP problems

Logfiles

When having issues the first step is to check the logfiles provided by PHP, the Web server and ownCloud itself.

Note: In the following the paths to the logfiles of a default Debian installation running Apache2 with mod_php is assumed. On other Web servers, Linux distros or operating systems they can differ.

- The logfile of Apache2 is located in /var/log/apache2/error.log.
- The logfile of PHP can be configured in your /etc/php5/apache2/php.ini. You need to set the directive log_errors to On and choose the path to store the logfile in the error_log directive. After those changes you need to restart your Web server.
- The logfile of ownCloud is located in the data directory /var/www/owncloud/data/owncloud.log.

Web server and PHP modules

Note: Lighttpd is not supported with ownCloud, and some ownCloud features may not work at all on Lighttpd.

There are some Web server or PHP modules which are known to cause various problems like broken up-/downloads. The following shows a draft overview of these modules:

- 1. Apache
- mod_pagespeed
- mod_evasive
- mod_security
- mod_reqtimeout
- mod_deflate
- libapache2-mod-php5filter (use libapache2-mod-php5 instead)
- mod_spdy together with libapache2-mod-php5 / mod_php (use fcgi or php-fpm instead)
- mod_dav
- mod_xsendfile / X-Sendfile (causing broken downloads if not configured correctly)
- 2. NginX
- ngx_pagespeed
- HttpDavModule
- X-Sendfile (causing broken downloads if not configured correctly)
- 3. PHP
- eAccelerator

11.1.4 Troubleshooting WebDAV

ownCloud uses SabreDAV, and the SabreDAV documentation is comprehensive and helpful.

See:

- SabreDAV FAQ
- Web servers (Lists lighttpd as not recommended)
- Working with large files (Shows a PHP bug in older SabreDAV versions and information for mod_security problems)
- 0 byte files (Reasons for empty files on the server)
- Clients (A comprehensive list of WebDAV clients, and possible problems with each one)
- Finder, OS X's built-in WebDAV client (Describes problems with Finder on various Web servers)

There is also a well maintained FAQ thread available at the ownCloud Forums which contains various additional information about WebDAV problems.

11.1.5 Troubleshooting Contacts & Calendar

Service discovery

Some clients - especially on iOS/Mac OS X - have problems finding the proper sync URL, even when explicitly configured to use it.

If you want to use CalDAV or CardDAV clients together with ownCloud it is important to have a correct working setup of the following URLs:

```
https://example.com/.well-known/carddav
https://example.com/.well-known/caldav
```

Those need to be redirecting your clients to the correct DAV endpoints. If running ownCloud at the document root of your Web server the correct URL is:

https://example.com/remote.php/dav

and if running in a subfolder like owncloud:

https://example.com/owncloud/remote.php/dav

For the first case the .htaccess file shipped with ownCloud should do this work for your when running Apache. You only need to make sure that your Web server is using this file. When running NGINX please refer to *nginx Example Configurations*.

If your ownCloud instance is installed in a subfolder called owncloud and you're running Apache create or edit the .htaccess file within the document root of your Web server and add the following lines:

Redirect 301 /.well-known/carddav /owncloud/remote.php/dav Redirect 301 /.well-known/caldav /owncloud/remote.php/dav

Now change the URL in the client settings to just use:

https://example.com

instead of e.g.

https://example.com/owncloud/remote.php/dav/principals/username.

There are also several techniques to remedy this, which are described extensively at the Sabre DAV website.

Unable to update Contacts or Events

If you get an error like:

PATCH https://example.com/remote.php/dav HTTP/1.0 501 Not Implemented

it is likely caused by one of the following reasons:

- Using Pound reverse-proxy/load balancer As of writing this Pound doesn't support the HTTP/1.1 verb. Pound is easily patched to support HTTP/1.1.
- **Misconfigured Web server** Your Web server is misconfigured and blocks the needed DAV methods. Please refer to *Troubleshooting WebDAV* above for troubleshooting steps.

11.1.6 Client Sync Stalls

One known reason is stray locks. These should expire automatically after an hour. If stray locks don't expire (identified by e.g. repeated file.txt is locked and/or Exception\\\\FileLocked messages in your data/owncloud.log), make sure that you are running system cron and not Ajax cron (See *Background Jobs*). See https://github.com/owncloud/core/issues/22116 and https://central.owncloud.org/t/file-is-locked-how-to-unlock/985 for some discussion and additional info of this issue.

11.1.7 Other issues

Some services like *Cloudflare* can cause issues by minimizing JavaScript and loading it only when needed. When having issues like a not working login button or creating new users make sure to disable such services first.

11.2 Code Signing

ownCloud supports code signing for the core releases, and for ownCloud applications. Code signing gives our users an additional layer of security by ensuring that nobody other than authorized persons can push updates.

It also ensures that all upgrades have been executed properly, so that no files are left behind, and all old files are properly replaced. In the past, invalid updates were a significant source of errors when updating ownCloud.

11.2.1 FAQ

Why Did ownCloud Add Code Signing?

By supporting Code Signing we add another layer of security by ensuring that nobody other than authorized persons can push updates for applications, and ensuring proper upgrades.

Do We Lock Down ownCloud?

The ownCloud project is open source and always will be. We do not want to make it more difficult for our users to run ownCloud. Any code signing errors on upgrades will not prevent ownCloud from running, but will display a warning on the Admin page. For applications that are not tagged "Official" the code signing process is optional.

Not Open Source Anymore?

The ownCloud project is open source and always will be. The code signing process is optional, though highly recommended. The code check for the core parts of ownCloud is enabled when the ownCloud release version branch has been set to stable.

For custom distributions of ownCloud it is recommended to change the release version branch in version.php to something else than "stable".

Is Code Signing Mandatory For Apps?

Code signing is optional for all third-party applications. Applications with a tag of "Official" on apps.owncloud.com require code signing.

11.2.2 Fixing Invalid Code Integrity Messages

A code integrity error message ("There were problems with the code integrity check. More information...") appears in a yellow banner at the top of your ownCloud Web interface:

There were problems with the code integrity check. More information...

Note: The yellow banner is only shown for admin users.

Clicking on this link will take you to your ownCloud admin page, which provides the following options:

- 1. Link to this documentation entry.
- 2. Show a list of invalid files.
- 3. Trigger a rescan.

There were problems with the code integrity check. More information...

Security & setup warnings

• Some files have not passed the integrity check. Further information on how to resolve this issue can be found in our documentation. (List of invalid files... / Rescan...)

To debug issues caused by the code integrity check click on "List of invalid files...", and you will be shown a text document listing the different issues. The content of the file will look similar to the following example:

Technical information

```
The following list covers which files have failed the integrity check. Please read the previous linked documentation to learn more about the errors and how to fix them.
Results
```

```
_____
- core
     - INVALID_HASH
              - /index.php
              - /version.php
      - EXTRA_FILE
              - /test.php
- calendar
      - EXCEPTION
              - OC\IntegrityCheck\Exceptions\InvalidSignatureException
              - Signature data not found.
- tasks
      - EXCEPTION
              - OC\IntegrityCheck\Exceptions\InvalidSignatureException
              - Certificate has been revoked.
Raw output
_____
Array
(
    [core] => Array
```

```
(
        [INVALID_HASH] => Array
            (
                [/index.php] => Array
                    (
                        [expected] =>
                        f1c5e2630d784bc9cb02d5a28f55d6f24d06dae2a0fee685f3
                        c2521b050955d9d452769f61454c9ddfa9c308146ade10546c
                        fa829794448eaffbc9a04a29d216
                        [current] =>
                        ce08bf30bcbb879a18b49239a9bec6b8702f52452f88a9d321
                        42cad8d2494d5735e6bfa0d8642b2762c62ca5be49f9bf4ec2
                        31d4a230559d4f3e2c471d3ea094
                    )
                [/version.php] => Array
                    (
                        [expected] =>
                        c5a03bacae8dedf8b239997901ba1fffd2fe51271d13a00cc4
                        b34b09cca5176397a89fc27381cbb1f72855fa18b69b6f87d7
                        d5685c3b45aee373b09be54742ea
                        [current] =>
                        88a3a92c11db91dec1ac3be0e1c87f862c95ba6ffaaaa3f2c3
                        b8f682187c66f07af3a3b557a868342ef4a271218fe1c1e300
                        c478e6c156c5955ed53c40d06585
                    )
            )
        [EXTRA_FILE] => Array
            (
                [/test.php] => Array
                    (
                        [expected] =>
                        [current] =>
                        09563164f9904a837f9ca0b5f626db56c838e5098e0ccc1d8b
                        935f68fa03a25c5ec6f6b2d9e44a868e8b85764dafd1605522
                        b4af8db0ae269d73432e9a01e63a
                    )
            )
   )
[calendar] => Array
    (
        [EXCEPTION] => Array
            (
                [class] => OC\IntegrityCheck\Exceptions\InvalidSignature
                Exception
                [message] => Signature data not found.
            )
   )
[tasks] => Array
    (
        [EXCEPTION] => Array
            (
```

```
[class] => OC\IntegrityCheck\Exceptions\InvalidSignatureException
[message] => Certificate has been revoked.
)
)
)
```

In above error output it can be seen that:

- 1. In the ownCloud core (that is, the ownCloud server itself) the files "index.php" and "version.php" do have the wrong version.
- 2. In the ownCloud core the unrequired extra file "/test.php" has been found.
- 3. It was not possible to verify the signature of the calendar application.
- 4. The certificate of the task application was revoked.

You have to do the following steps to solve this:

- 1. Upload the correct "index.php" and "version.php" files from e.g. the archive of your ownCloud version.
- 2. Delete the "test.php" file.
- 3. Contact the developer of the application. A new version of the app containing a valid signature file needs to be released.
- 4. Contact the developer of the application. A new version of the app signed with a valid signature needs to be released.

For other means on how to receive support please take a look at https://owncloud.org/support/. After fixing these problems verify by clicking "Rescan...".

Note: When using a FTP client to upload those files make sure it is using the Binary transfer mode instead of the ASCII transfer mode.

11.2.3 Rescans

Rescans are triggered at installation, and by updates. You may run scans manually with the occ command. The first command scans the ownCloud core files, and the second command scans the named app. There is not yet a command to manually scan all apps:

```
occ integrity:check-core
occ integrity:check-app $appid
```

See Using the occ Command to learn more about using occ.

11.2.4 Errors

Warning: Please don't modify the mentioned signature.json itself.

The following errors can be encountered when trying to verify a code signature.

- INVALID_HASH
 - The file has a different hash than specified within signature.json. This usually happens when the file has been modified after writing the signature data.

- MISSING_FILE
 - The file cannot be found but has been specified within signature.json. Either a required file has been left out, or signature.json needs to be edited.
- EXTRA_FILE
 - The file does not exist in signature.json. This usually happens when a file has been removed and signature.json has not been updated. It also happens if you have placed additional files in your ownCloud installation folder.
- EXCEPTION
 - Another exception has prevented the code verification. There are currently these following exceptions:
 - * Signature data not found.'
 - The app has mandatory code signing enforced but no signature.json file has been found in its appinfo folder.
 - * Certificate is not valid.
 - The certificate has not been issued by the official ownCloud Code Signing Root Authority.
 - * Certificate is not valid for required scope. (Requested: %s, current: %s)
 - The certificate is not valid for the defined application. Certificates are only valid for the defined app identifier and cannot be used for others.
 - * Signature could not get verified.
 - There was a problem with verifying the signature of signature.json.
 - * Certificate has been revoked.
 - \cdot The certificate which was used to sign the application was revoked.

CHAPTER

TWELVE

ENTERPRISE EDITION ONLY

12.1 Enterprise Edition Installation

12.1.1 Installing & Upgrading ownCloud Enterprise Edition

The recommended method for installing and maintaining your ownCloud Enterprise edition is with your Linux package manager. Configure your package manager to use the ownCloud Enterprise repository, import the signing key, and then install and update ownCloud packages like any other software package. Please refer to the README – ownCloud Package Installation.txt document in your account at Customer.owncloud.com account for instructions on setting up your Linux package manager.

After you have completed your initial installation of ownCloud as detailed in the README, follow the instructions in *Installation Wizard* to finish setting up ownCloud.

To upgrade your Enterprise server, refer to How to Upgrade Your ownCloud Server.

Manual Installation

Download the ownCloud archive from your account at https://customer.owncloud.com/owncloud, then follow the instructions at *Manual Installation on Linux*.

SELinux

Linux distributions that use SELinux need to take some extra steps so that ownCloud will operate correctly under SELinux. Please see *SELinux Configuration* for some recommended configurations.

12.1.2 Supported ownCloud Enterprise Edition Apps

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

Note: 3rd party and unsupported apps must be disabled before performing a system upgrade. Then install the upgraded versions, and after the upgrade is complete re-enable them.

12.1.3 License Keys

Introduction

You'll need to install a license key to use ownCloud Enterprise Edition. There are two types of license keys: one is a free 30-day trial key. The other is a full license key for Enterprise customers.

You can download and try ownCloud Enterprise for 30 days for free, which auto-generates a free 30-day key. When this key expires your ownCloud installation is not removed, so when you become an Enterprise customer you can enter your new key to regain access. See How to Buy ownCloud for sales and contact information.

Configuration

Once you get your Enterprise license key, it needs to be copied to your ownCloud configuration file, config/config.php file like this example:

Each running instance of ownCloud requires a license key. Keys will work across upgrades without issue, so new keys will not be required when you upgrade your ownCloud Enterprise to a new version.

12.1.4 Oracle Database Setup

This document will cover the setup and preparation of the ownCloud server to support the use of Oracle as a backend database. For the purposes of testing, we are using Oracle Enterprise Linux as both the Web server that will host ownCloud, and as a host for the Oracle Database.

Outline of Steps

This document will cover the following steps:

- Setup of the ownCloud user in Oracle: This involves setting up a user space in Oracle for setting up the own-Cloud database.
- Installing the Oracle Instant Client on the Web server (facilitating the connection to the Oracle Database).
- Compiling and installing the Oracle PHP Plugin oci8 module
- · Pointing ownCloud at the Oracle database in the initial setup process

The document assumes that you already have your Oracle instance running, and have provisioned the needed resources. It also assumes that you have installed ownCloud with all of the prerequisites.

Configuring Oracle

Setting up the User Space for ownCloud

Step one, if it has not already been completed by your DBA (DataBase Administrator), provision a user space on the Oracle instance for ownCloud. This can be done by logging in as a DBA and running the script below:

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

Substitute an actual password for password. Items like TableSpace, Quota etc. will be determined by your DBA.

Downloading and Installing the Oracle Instant Client

As our example system is Oracle Enterprise Linux, it is necessary to go to the Oracle site and download the Oracle Instant Client for your OS Distribution.

Note: Download the instant client and the instant client SDK and place them in a directory on the server, in this example they are RPM packages.

- Install the basic client from the RPM. Use the rpm -ivh command
- Install the SDK RPM package. Use the rpm -ivh command

At this point, the Oracle Instant client is installed on the ownCloud Host (in the home directory).

Install the OCI8 PHP Extension:

The next step is to compile and install the OCI8 PHP extension for connectivity to the Oracle Database.

- Create a folder for these bits on your server.
- Download the latest version of the extension from http://pecl.php.net/package/oci8.
- Unpack the OCI8 PHP extension and copy it over to the server.
- There should be two things in the folder:
 - package.xml file
 - oci8-*.*.* folder (folder will change based on version of the extension you downloaded).
- Build the OCI8 module.
 - Change (cd) to the folder where you have copied the downloaded and uncompressed OCI8 bits.
 - Run the following command (there will be a significant amount of output):

pecl build

- Eventually the output will stop and ask for the Oracle Home Directory, just press enter.
- Change directory:

cd oci8-<version number>

• Type the following command:

./configure -with-oci8=instantcleint,/usr/lib/oracle/<version number>/client64/lib

- · Again, there will be significant output
- Enter the following command to compile: make
- At this time there should be a folder called modules in the oci8-<version_> folder. Within this folder exists the oci8.so file.
- Copy this to the directory where the modules are stored in the PHP install. It depends on your distribution. This is the path for RHEL 6 and OEL 6:

cp oci8.so /usr/lib64/php/modules

- Create an . ini file
 - Navigate to the php.d directory: cd /etc/php.d

- Edit a file called oci8.ini: vi oci8.ini
- Make the file look as follows:

; Oracle Instant Client Shared Object extension=oci8.so

- Save the document

Configure ownCloud

The next step is to configure the ownCloud instance to point to the Oracle Database, again this document assumes that ownCloud has previously been installed.

Configuration Wizard

 Usemame Password Advanced Advanced Data folder //var/www/owncloud/data /var/www/owncloud/data Configure the database Oracle will be used. Database user Database password Database name Database tablespace localhost	Create an admin account	
••• Password • Advanced • Data folder /var/www/owncloud/data Configure the database Oracle will be used. Database user Database password Database name Database tablespace localhost	1 Username	
Advanced V Data folder /var/www/owncloud/data Configure the database Oracle will be used. Database user Database password Database name Database tablespace	··· Password	0
Data folder /var/www/owncloud/data Configure the database Oracle will be used. Database user Database password Database name Database tablespace localhost	Advanced V	
<pre>/var/www/owncloud/data Configure the database Oracle will be used.</pre> Database user Database password Database name Database tablespace localhost	Data folder	
Configure the database Oracle will be used. Database user Database password Database name Database tablespace localhost	/var/www/owncloud/data	
Database user Database password Database name Database tablespace localhost	Configure the database Oracle will be used.	
Database password Database name Database tablespace localhost		
Database name Database tablespace localhost	Database user	
Database tablespace	Database user Database password	
localhost	Database user Database password Database name	
	Database user Database password Database name Database tablespace	

Database user This is the user space created in step 2.1. In our Example this would be owncloud.

Database password Again this is defined in the script from section 2.1 above, or pre-configured and provided to you by your DBA.

Database Name Represents the database or the service that has been pre-configured on the TSN Listener on the Database Server. This should also be provided by the DBA. In this example, the default setup in the Oracle install was orcl (there is a TSN Listener entry for orcl on our database server).

This is not like setting up with MySQL or SQL Server, where a database based on the name you give is created. The oci8 code will call this specific service and it must be active on the TSN Listener on your Oracle Database server.

Database Table Space Provided by the DBA. In this example the users table space (as is seen in the user creation script above), was used.

Configuration File

Assuming all of the steps have been followed to completion, the first run wizard should complete successfully, and an operating instance of ownCloud should appear.

The configuration file should look something like this:

```
<?php
$CONFIG = array (
'instanceid' => 'abcdefgh',
'passwordsalt' => '01234567890123456789',
'datadirectory' => '/var/data',
'dbtype' => 'oci',
'version' => '8.2.x.y',
'dbname' => 'orcl',
'dbhost' => '192.168.1.57',
'dbtableprefix' => 'oc_',
'dbtableprefix' => 'oc_',
'dbpassword' => '*******',
'installed' => true,
);
```

Useful SQL Commands

Is my Database Reachable?

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

```
sqlplus username
```

SQL> select * from v\$version;

BANNER

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

SQL> exit

Show Database Users:

Oracle : SELECT * FROM all_users;

Show available Databases:

Oracle : SELECT name FROM v\$database; (requires DBA privileges)

Show ownCloud Tables in Database:

Oracle : SELECT table_name FROM user_tables;

Quit Database:

Oracle : quit

12.2 Creating Branded ownCloud Clients (Enterprise only)

12.2.1 Creating Branded Client Apps (Enterprise Only)

Overview

ownBrander is an ownCloud build service that is exclusive to Enterprise customers for creating branded Android and iOS ownCloud sync apps, and branded ownCloud desktop sync clients. You build your apps with the ownBrander app on your Customer.owncloud.com account, and within 24-48 hours the completed, customized apps are loaded into your account. You must supply your own artwork, and you'll find all the specifications and required elements in ownBrander.



Building a Branded Desktop Sync Client

See Building Branded ownCloud Clients (Enterprise Only) for instructions on building your own branded desktop sync client, and for setting up an automatic update service.

Your users may run both a branded and un-branded desktop sync client side-by-side. Both clients run independently of each other, and do not share account information or files.

Building a Branded iOS App

Building and distributing your branded iOS ownCloud app involves a large number of interdependent steps. The process is detailed in the Building Branded ownCloud Clients (Enterprise Only) manual. Follow these instructions exactly and in order, and you will have a nice branded iOS app that you can distribute to your users.

Building an Android App

Building and distributing your branded Android ownCloud app is fairly simple, and the process is detailed in Building Branded ownCloud Clients (Enterprise Only).

12.2.2 Custom Client Download Repositories

See *Custom Client Download Repositories* to learn how test and configure custom download repository URLs for your branded clients.

12.3 Enterprise Server Branding (Enterprise only)

12.3.1 Custom Theming ownCloud (Enterprise only)

Overview

ownBrander is an ownCloud build service that is exclusive to Enterprise edition customers for creating branded own-Cloud clients and servers. You may brand your ownCloud server using ownBrander to easily build a custom theme, using your own logo and artwork. ownCloud has always been theme-able, but it was a manual process that required editing CSS and PHP files. Now Enterprise customers can use ownBrander, which provides an easy graphical wizard.

You need an Enterprise subscription, an account on Customer.owncloud.com, and the ownBrander app enabled on your account. When you complete the steps in the wizard the ownBrander service builds your new branded theme, and in 24-48 hours you'll see it in your account.



When you open the ownBrander app, go to the Web tab. You will see an introduction and the wizard, which starts with uploading your logo. You will need a number of images in specific sizes and formats, and the wizard tells you what you need. Example images are on the right, and you can click to enlarge them.

Note: If you see errors when you upload SVG files, such as "Incorrect extension.File type image/svg+xml is not correct", "This SVG is invalid", or "Error uploading file: Incorrect size", try opening the file in Inkscape then save as "Plain SVG" and upload your SVG image again.

The wizard has two sections. The first section contains all the required elements: logos and other artwork, colors, naming, and your enterprise URL. The Suggested section contains optional items such as additional logo placements and custom URLs.

When you are finished, click the **Generate Web Server** button. If you want to change anything, go ahead and change it and click the **Generate Web Server** button. This will override your previous version, if it has not been created yet. In 24-48 hours you'll find your new branded theme in the **Web** folder in your Customer.owncloud.com account.

Inside the **Web** folder you'll find a **themes** folder. Copy this to your owncloud/themes directory. You may name your **themes** folder anything you want, for example myBrandedTheme. Then configure your ownCloud server to use your branded theme by entering it in your config.php file:

```
"theme" => "myBrandedTheme"
```

Login logo images



Login page logo

This is the image shown on the login page just above the Username and Password fields (svg format) (width: 252px height: 122px) *i*



Upload



Login page logo

This is the image shown on the login page just above the Username and Password fields. This image is used when the browser does not support svg, we recommend this to be the same as the

previous one (Login page logo) (png format) (width: 252px height: 122px) i



Logo icon



If anything goes wrong with your new theme, comment out this line to re-enable the default theme until you fix your branded theme. The branded theme follows the same file structure as the default theme, and you may further customize it by editing the source files.

Note: Always edit only your custom theme files. Never edit the default theme files.

12.4 External Storage (Enterprise only)

12.4.1 Enterprise-Only Authentication Options

In ownCloud 9.0+, there are five authentication backends for external storage mounts:

- · Username and password
- · Log-in credentials, save in session
- Log-in credentials, save in database
- User entered, store in database
- Global credentials

The first two are common to all editions of ownCloud, and the last three are only in the Enterprise edition. These are available to:

- FTP
- ownCloud
- SFTP
- SMB/CIFS
- WebDAV
- Windows Network Drive
- **Username and password** This is the default; a login entered by the admin when the external mount is created. The login is stored in the database, which allows sharing, and background jobs, such as file scanning, to operate.
- Log-in credentials, save in session Credentials are only stored in the session and not captured in the database. Files cannot be shared, as credentials are not stored.

Log-in credentials, save in database Credentials are stored in the database, and files can be shared.

User entered, store in database Users provide their own login credentials, rather than using admin-supplied credentials. User credentials are stored in the database, and files can be shared.

Global credentials Re-usable credentials entered by the admin, files can be shared.

Global credentials are entered in a separate form.

Use the dropdown selector to choose the authentication backend when you create a new external mount.

12.4.2 LDAP Home Connector

The LDAP Home Connector App enables you to configure your ownCloud server to display your users' Windows home directories on their Files pages, just like any other folder. Typically, Windows home directories are stored on a network server in a root folder, such as Users, which then contains individual folders for each user.

External Storage

Global credentials for external storages

Username	Password	Save
Username and password -		
Username and password		
Log-in credentials, save in session		
Log-in credentials, save in database		
User entered, store in database		
Global Credentails		

You must already have the LDAP app enabled and a working LDAP/Active Directory configuration in ownCloud.

Next, configure the root Windows home directory to be mounted on your ownCloud server. Then use the LDAP Home Connector and LDAP app to connect it to ownCloud.

Mount Home Directory

Create an entry in /etc/fstab for the remote Windows root home directory mount. Store the credentials to access the home directory in a separate file, for example /etc/credentials, with the username and password on separate lines, like this:

username=winhomeuser password=winhomepassword

Then add a line like this to /etc/fstab, substituting your own server address and filenames:

//192.168.1.58/share /mnt/share cifs credentials=/etc/credentials,uid=33,gid=33

Configure the LDAP Home Connector

Enable the LDAP Home Connector app. Then go to the LDAP Home Connector form on your ownCloud admin page. In the **Display folder as:** field enter the name as you want it to appear on your users' File pages.

Then in the **Attribute name:** field enter the LDAP attribute name that will contain the home directory. Use any LDAP attribute that is not already in use, then save your changes.

LDAP User Home

Display folder as:	Windows Home Directory
Attribute name:	userSharedFolder
Save	

Configure the LDAP Server

In Active Directory, open the user profile. Scroll to the Extensions section and open the Attribute Editor tab

Remote control Remote Desktop Services P Personal Virtual Desktop Security I Published Certificates Password Replication Attribute Attribute Value Attribute accountExpires (never) accountNameHistory <not set=""> adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> altSecurityIdentities <not set=""> audio <not set=""> audio <not set=""> audio <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not></not></not></not>	ions	Sessions	nment	Enviro	COM+
Personal Virtual Desktop Security Published Certificates Password Replication Attribute Attribute Value Attribute accountExpires (never) accountExpires accountExpires adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> altSecurityIdentities <not set=""> audio <not set=""> audio <not set=""> audio <not set=""> adminDisplayName <not set=""> audio <not set=""> audio <not set=""> audio <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""></not></not></not></not></not></not></not></not></not></not></not></not></not>	rofile	ervices Profile	Remote Desktop		Remote control
Published Certificates Password Replication Attribute Attribute Value accountExpires (never) accountNameHistory <not set=""> adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not></not></not></not></not></not>	Dial-in	Dial-i	Security	top	Personal Virtual Desk
Attribute Value accountExpires (never) accountNameHistory <not set=""> aCSPolicyName <not set=""> adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not></not></not>	ute Edito	Attribute E	ord Replication	Passw	ublished Certificates
Attribute Value accountExpires (never) accountNameHistory <not set=""> aCSPolicyName <not set=""> adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not></not></not>					tributes:
accountExpires (never) accountNameHistory <not set=""> aCSPolicyName <not set=""> adminCount <not set=""> adminDescription <not set=""> adminDisplayName <not set=""> altSecurityIdentities <not set=""> assistant <not set=""> audio <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not></not></not></not>				Value	- Attribute
accountNameHistory <not set="">aCSPolicyName<not set="">adminCount<not set="">adminDescription<not set="">adminDisplayName<not set="">altSecurityIdentities<not set="">assistant<not set="">attributeCertificateAttri<not set="">audio<not set="">badPasswordTime(never)badPwdCount0businessCategory<not set="">c<not set=""></not></not></not></not></not></not></not></not></not></not></not>				(never)	accountExpires
aCSPolicyName <not set="">adminCount<not set="">adminDescription<not set="">adminDisplayName<not set="">altSecurityIdentities<not set="">assistant<not set="">attributeCertificateAttri<not set="">audio<not set="">badPasswordTime(never)badPwdCount0businessCategory<not set="">c<not set=""></not></not></not></not></not></not></not></not></not></not>				<not set=""></not>	accountNameHistory
adminCount <not set="">adminDescription<not set="">adminDisplayName<not set="">altSecurityIdentities<not set="">assistant<not set="">attributeCertificateAttri<not set="">audio<not set="">badPasswordTime(never)badPwdCount0businessCategory<not set="">c<not set=""></not></not></not></not></not></not></not></not></not>				<not set=""></not>	aCSPolicyName
adminDescription <not set=""> adminDisplayName <not set=""> altSecurityIdentities <not set=""> assistant <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not></not>				<not set=""></not>	adminCount
adminDisplayName <not set=""> altSecurityIdentities <not set=""> assistant <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not></not>				<not set=""></not>	adminDescription
alt SecurityIdentities <not set=""> assistant <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not></not>				<not set=""></not>	admin DisplayName
assistant <not set=""> attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not></not>				<not set=""></not>	altSecurityIdentities
attributeCertificateAttri <not set=""> audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not></not>				<not set=""></not>	assistant
audio <not set=""> badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not></not>				<not set=""></not>	attributeCertificateAttri
badPasswordTime (never) badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not>				<not set=""></not>	audio
badPwdCount 0 businessCategory <not set=""> c <not set=""></not></not>				(never)	badPasswordTime
businessCategory <not set=""> c <not set=""></not></not>				0	badPwdCount
c <not set=""></not>				<not set=""></not>	businessCategory
	-			<not set=""></not>	C
•)			(
	_				1. A.
Edit	Filter	Filter			Edit

Scroll to the attribute being used (UserSharedFolder in this instance), and click Edit. Enter the users home directory.

String Attri	ibute Editor	×	
Attribute:	userSharedFolder	r _	
Value:			
/mnt/s	hare/Users	pr	
,,			
Clear		OK Cancel	Î
	unixHomeDirectory	<not set=""></not>	
	unixUserPassword	<not set=""></not>	
	hu	<not set=""></not>	
	userAccountControl	0x10200 = (NORMAL_ACCOUNT DONT_E	
	userCert	<not set=""></not>	
	userCertificate	<not set=""></not>	
	userParameters	<not set=""></not>	
	userPassword	<not set=""></not>	
	userPKCS12	<not set=""></not>	
	userPrincipalName	steve@owncloud1.com	
	userSharedFolder	/mnt	
	userSharedFolderOther	<not set=""></not>	
	userSMIMECertificate	<not set=""></not>	
	userWorkstations	<not set=""></not>	
	•		
	<u>E</u> dit	<u>F</u> ilter	
			-

Save your changes, and you are finished.

12.4.3 Configuring SharePoint Integration

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

The ownCloud SharePoint plugin uses SharePoint document lists as remote storage folders. ownCloud respects SharePoint access control lists (ACLs), so ownCloud sharing is intentionally disabled for SharePoint mountpoints. This is to preserve SharePoint ACLs and ensure content is properly accessed as per SharePoint rules.

The plugin uses the Simple Object Access Protocol (SOAP) and WebDAV for the uploads and downloads to talk to SharePoint servers. Your ownCloud server must have php-soap or php5-soap installed. Linux packages and ownCloud appliances will install php5-soap as a required dependency.

The supported authentication methods are:

- Basic Auth
- NTLM (Recommended)

Creating a Sharepoint Mount

Enable the Sharepoint app, and then enter the Admin panel to set up SharePoint connections in the SharePoint Drive Configuration section.

Enter your SharePoint Listing credentials. These credentials are not stored in the database, but are used only during plugin setup to list the Document Libraries available per SharePoint site.

SharePoint Configuration

Listing credentials. These fields are only used to list available SharePoint document list. They are not stored.

administrator		
Global credentials.	These fields can be used	for each of the SharePoint mounts
administrator		

Global credentials is optional. If you fill in these fields, these credentials will be used on on all SharePoint mounts where you select: Use global credentials as the authentication credentials.

Local Folder Name	Available for	SharePoint Site Url	Document Library
sharepoint 1	All users	https://example.com	folder1
sharepoint2	All users	https://example2.com	folder2

Enter your ownCloud mountpoint in the Local Folder Name column. This is the name of the folder that each user will see on the ownCloud filesystem. You may use an existing folder, or enter a name to create a new mount point

Select who will have access to this mountpoint, by default **All users**, or a user or a group.

Enter your SharePoint server URL, then click the little refresh icon to the left of the Document Library field. If your credentials and URL are correct you'll get a dropdown list of available SharePoint libraries. Select the document library you want to mount.

Authentication credentials



Select which kind of Authentication credentials you want to use for this mountpoint. If you select **Custom credentials** you will have to enter the the credentials on this line. Otherwise, the global credentials or the user's own credentials will be used. Click Save, and you're done

Enabling Users

You may allow your users to create their own Sharepoint mounts on their Personal pages, and allow sharing on these mounts.

✓ Allow users to mount their own SharePoint document libraries

Allow users to share content in SharePoint mount points

Note

Speed up load times by disabling file previews in config.php, because the previews are generated by downloading the remote files to a temp file. This means ownCloud will spend a lot of time creating previews for all of your SharePoint content. To disable file previews, add the following line to the ownCloud config file found in /owncloud/config/config.php:

'enable_previews' => false,

Troubleshooting

SharePoint unsharing is handled in the background via Cron. If you remove the sharing option from a Sharepoint mount, it will take a little time for the share to be removed, until the Cron job runs

Turn on Sharepoint app logging by modifying the following line in apps/sharepoint/lib/sharepoint.php to TRUE:

private static \$enableLogs = TRUE;

Global mount points can't be accessed: You have to fill out your SharePoint credentials as User on the personal settings page, or in the popup menu. These credentials are used to mount all global mount points.

Personal mount points can't be accessed: You have to fill your SharePoint credentials as User on the personal settings page in case your personal mount point doesn't have its own credentials.

A user can't update the credentials: Verify that the correct credentials are configured, and the correct type, either global or custom.

12.4.4 Installing and Configuring the Windows Network Drive App

The Windows Network Drive app creates a control panel on your Admin page for seamless mounting of SMB/CIFS file shares on ownCloud servers.

Any Windows file share, and Samba servers on Linux and other Unix-type operating systems use the SMB/CIFS file-sharing protocol. The files and directories on the SMB/CIFS server will be visible on your Files page just like your other ownCloud files and folders. They are labeled with a little four-pane Windows-style icon, and the left pane of your Files page includes a Windows Network Drive filter. Figure 1 shows a new Windows Network Drive share marked with red warnings. These indicate that ownCloud cannot connect to the share because it requires the user to login, it is not available, or there is an error in the configuration.

Files are synchronized bi-directionally, and you can create, upload, and delete files and folders. ownCloud server admins can create Windows Network Drive mounts, and optionally allow users to create their own personal Windows Network Drive mounts.



Figure 12.1: Figure 1: Windows Network Drive share on your Files page.

Depending on the authentication method, passwords for each mount are encrypted and stored in the ownCloud database, using a long random secret key stored in config.php, which allows ownCloud to access the shares when the users who own the mounts are not logged in. Or, passwords are not stored and available only for the current session, which adds security.

Installation

Enable the Windows Network Drive app on your ownCloud Apps page. Then there are a few dependencies to install.

You must install php-smbclient. This should be included in most Linux distributions. See eduardok/libsmbclient-php if your distribution does not include it; this provides source archives and instructions for installing binary packages.

You also need the Samba client installed on your Linux system. This is included in all Linux distributions; on Debian, Ubuntu, and other Debian derivatives this is smbclient. On SUSE, Red Hat, CentOS, and other Red Hat derivatives it is samba-client.

Creating a New Share

When you create a new WND share you need the login credentials for the share, the server address, the share name, and the folder you want to connect to.

- 1. First enter the ownCloud mountpoint for your new WND share. This must not be an existing folder.
- 2. Then select your authentication method; See *Enterprise-Only Authentication Options* for complete information on the five available authentication methods.

	Folder name	External storage	Authentication
•	WND	Windows Network Drive	Username and password ✓ Username and password Log-in credentials, save in session Log-in credentials, save in database User entered, store in database Global Credentails

Figure 12.2: Figure 2: WND mountpoint and authorization credentials.

- 3. Enter the address of the server that contains the WND share.
- 4. The Windows share name.
- 5. The root folder of the share. This is the folder name, or the *suser* variable for user's home directories. Note that the LDAP Internal Username Attribute must be set to the samaccountname for either the share or the root to work, and the user's home directory needs to match the samaccountname. (See User Authentication with LDAP.)
- 6. Login credentials.
- 7. Select users or groups with access to the share. The default is all users.

8. Click the gear icon for additional mount options. Note that encryption is enabled by default, while sharing is not. Sharing is not available for all authorization methods; see *Enterprise-Only Authentication Options*.



Figure 12.3: Figure 3: WND server, credentials, and additional mount options.

Your changes are save automatically.

Note: When you create a new mountpoint using Login credentials you must log out of ownCloud, and then log back in so you can access the share. You only have to do this the first time.

Personal WND Mounts

Users create their own personal WND mounts on their Personal pages. These are created the same way as Admincreated shares. Users have four options for login credentials:

- Username and password
- Log-in credentials, save in session
- Log-in credentials, save in database
- · Global credentials

libsmclient Issues

If your Linux distribution ships with libsmbclient 3.x, which is included in the Samba client, you may need to set up the HOME variable in Apache to prevent a segmentation fault. If you have libsmbclient 4.1.6 and higher it doesn't seem to be an issue, so you won't have to change your HOME variable.

To set up the HOME variable on Ubuntu, modify the /etc/apache2/envvars file:

```
unset HOME
export HOME=/var/www
```

In Red Hat/CentOS, modify the /etc/sysconfig/httpd file and add the following line to set the HOME variable in Apache:

export HOME=/usr/share/httpd

By default CentOS has activated SELinux, and the httpd process can not make outgoing network connections. This will cause problems with the curl, ldap and samba libraries. You'll need to get around this in order to make this work. First check the status:

```
getsebool -a | grep httpd
httpd_can_network_connect --> off
```

Then enable support for network connections:

setsebool -P httpd_can_network_connect 1

In openSUSE, modify the /usr/sbin/start_apache2 file:

```
export HOME=/var/lib/apache2
```

Restart Apache, open your ownCloud Admin page and start creating SMB/CIFS mounts.

12.4.5 Configuring S3 and OpenStack Swift Objects as Primary Storage

In ownCloud Enterprise Subscription, you can configure S3 objects as primary storage. This replaces the default own-Cloud owncloud/data directory. You may still need to keep the owncloud/data directory for these reasons:

- The ownCloud log file is saved in the data directory
- Legacy apps may not support using anything but the owncloud/data directory

You can move your logfile by changing its location in config.php. You may still need owncloud/data for backwards compatibility with some apps.

Implications

ownCloud in object store mode expects exclusive access to the object store container, because it only stores the binary data for each file. The metadata are kept in the local database for performance reasons.

The current implementation is incompatible with any app that uses direct file I/O and circumvents the ownCloud virtual filesystem. That includes Encryption and Gallery. Gallery stores thumbnails directly in the filesystem, and Encryption causes severe overhead because key files need to be fetched in addition to any requested file.

Configuration

Look in config.sample.php for a example configurations. Copy the relevant part to your config.php file. Any object store needs to implement \\OCP\\Files\\ObjectStore\\IObjectStore and can be passed parameters in the constructor with the arguments key:

```
'objectstore' => [
    'class' => 'Implementation\\Of\\OCP\\Files\\ObjectStore\\IObjectStore',
    'arguments' => [
    ...
```

],],

Amazon S3

The S3 backend mounts a bucket of the Amazon S3 object store into the virtual filesystem. The class to be used is OCA\ObjectStore\S3:

```
'objectstore' => [
    'class' => 'OCA\ObjectStore\S3',
    'arguments' => [
        // replace with your bucket
        'bucket' => 'owncloud',
        'autocreate' => true,
        // uncomment to enable server side encryption
        //'serversideencryption' => 'AES256',
        'options' => [
            // version and region are required
            'version' => '2006-03-01',
            // change to your region
            'region' => 'eu-central-1',
            'credentials' => [
                // replace key and secret with your credentials
                'key' => 'EJ39ITYZEUH5BGWDRUFY',
                'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
            ],
       ],
    ],
],
```

Ceph S3

The S3 backend can also be used to mount the bucket of a ceph object store via the s3 API into the virtual filesystem. The class to be used is OCA\ObjectStore\S3:

```
'objectstore' => [
   'class' => 'OCA\ObjectStore\S3',
    'arguments' => [
       // replace with your bucket
       'bucket' => 'owncloud',
        'autocreate' => true,
        'options' => [
            // version and region are required
            'version' => '2006-03-01',
           'region' => '',
            // replace key, secret and bucket with your credentials
            'credentials' => [
                // replace key and secret with your credentials
                'kev'
                      => 'EJ39ITYZEUH5BGWDRUFY',
                'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
            ],
            // replace the ceph endpoint with your rgw url
            'endpoint' => 'http://cephhost:8000/',
            // Use path style when talking to ceph
            'command.params' => [
```

```
'PathStyle' => true,
],
],
],
],
```

OpenStack Swift

The Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem. The class to be used is \\OC\\Files\\ObjectStore\\Swift:

```
'objectstore' => [
    'class' => 'OC\\Files\\ObjectStore\\Swift',
    'arguments' => [
        'username' => 'demo',
        'password' => 'password',
        'container' => 'owncloud',
        'autocreate' => true,
        'region' => 'RegionOne',
        'url' => 'http://devstack:5000/v2.0',
        'tenantName' => 'demo',
        'serviceName' => 'swift',
        // url Type, optional, public, internal or admin
        'urlType' => 'internal'
],
```

12.4.6 Jive Integration

The Jive application allows Jive users to access files stored in Jive from a mobile device, tablet, or desktop client. Users have complete access through ownCloud Enterprise edition to upload, edit or download their files.

Jive can be configured as a data storage location for ownCloud, which means files saved in Jive appear in folders within ownCloud. Jive remains the system of record while ownCloud acts as a proxy, providing end-to-end file access for users at their desks and on the go.

Configuration

The Jive application is installed under the owncloud/apps directory on the server and enabled via the ownCloud admin screen. This app is only available for ownCloud EE v6 or higher. Go to the ownCloud admin screen section "Jive backend parameters" to configure the app to match your Jive server system parameters.

Jive backend parameters
Server Settings
Verify https certificate Verify the Jive https server certificate. Certificate must be installed in the system
Authentication mechanism to use against Jive basic 🔻
Jive api url http://owncloudlex.no-ip.org:8080/api/core/v3/ URL pointing to the Jive REST API V3. (https://mycompany.jiveon.com/api/core/v3/)
Filters
Jive category filter List of categories that files must have to be shown. Only applied for groups and files inside those groups, not for private files. Leave empty to not filter (HopBox)
Jive category separator ; Use this char to separate the list of categories. A comma by default (,)
Jive tag filter Tag to use ONLY for private stuff in Jive. This won't be used for groups or files inside groups. Leave empty to not filter (myhopbox)
Jive forbidden extensions exe,zip List of forbidden extensions (.exe,.zip,.tar.bz)
Jive forbidden extensions separator , Use this char to separate the list of extensions (,)
Jive maximum upload filesize 25 Maximum filesize allowed in MB (25)
Show groups of which you are a member If not checked, the plugin will show all available groups for you matching the filter, even groups that you are not a member
Jive FS mount point Jive File Share Folder where the Jive FS will be mounted. (Ribbit)
Jive private folder My Jive Folder name for private stuff in Jive (My HopBox)
Activate large file sharing for Jive Activate large file sharing subsytem
Jive large sharing FS mount point Too Big For Folder where the Jive large sharing FS will be mounted. (Too Big For)
Miscellaneous
Notification time for the connectivity check 10 Number of seconds that the notification (if any) for the connectivity check will last (30)
Cave

Parameter	Description	Values
Https	Verify the https server certificate. Certificate must be installed on the	Checkbox –
-	system.	enabled/disable
Authentication	Chose the Authentication mechanism to use against Jive	basic OR oAuth
Jive api url	URL string pointing to the Jive API	Example:
-		https://mycompany.jiveon.com/api/core/
Jive FS mount	Folder where the Jive File share will be mounted	String value up to 10
point		characters max
Jive category	List of categories that files have to be shown	Jive categories list, or
filter		blank
Jive category	Separator for Jive catagories list	Comma by default or
separator		any single character
Jive tag filter	Tag to use for private stuff in jive	Jive tag or blank
Jive forbidden	List of forbidden extensions	Examples include:
extensions	These will not be allowed for upload or download with Jive.	.exe,.zip
Jive forbidden	Use this character to separate the list of extensions	Comma by default or
extensions		any single character
separator		
Jive maximum	Maximum file size allowed in MB. This includes upload and	Numeric value
upload filesize	downloads.	
Jive private	Folder name for private stuff in Jive	String value up to
folder		250 characters max
Activate large	Enable the large file sharing subsystem. This allows storage of files	Checkbox –
file sharing for	that are too large for Jive to be stored on the ownCloud server and	enable/disable
Jive	available via the ownCloud web, mobile and desktop interfaces.	
Jive large file	Folder where the Jive large sharing File Share will be mounted	String value up to 10
sharing FS		characters max
mount point		
Show groups of	If this is not checked, the plugin will show all available groups for	Enable/disable
which you are a	you matching the filter, even groups that you are not a member	
member		

Use Cases

The ownCloud Jive plugin can be used in various ways to extend the access to the Jive content across multiple devices.

Web Client Use Cases

Cloud Enterprise Edition			
[==	New	•	
Files		Name	
4 Activity		Jive File Share	
		Too Big For	
		2 folders	

- Create a folder in the "Jive File Share" Web Client folder to create a new Jive Group.
 - Verify the Group is created in Jive.
- Create a new Group in Jive and upload a file to that Group.
 - Check the Web Client and download the file.
 - Verify that file is the same as the uploaded file.
- Upload a file in the "Too Big For" Jive folder, and create the link in a Jive document.
 - Verify that file link is in Jive.
 - Download the file via the link, and verify it is the same as the uploaded file.
- Upload a file to the private "My Jive" Web Client folder.
 - Check your Jive content and make sure the file has been uploaded.
 - Download the file and verify it is the same as the uploaded file.

Mobile Client Use Cases (iOs and Android)

Create a new folder in the Mobile Client to create a new Jive Group.

Upload a file in the Web Client folder, and see that file in the corresponding Jive Group.

Desktop Client Use Cases

Create a folder in the Desktop Client to create a new Jive Group.

Upload a file in the Desktop Client folder, and see that file in the corresponding Jive Group.

The ownCloud folder structure hierarchy matches the Jive Groups the user can access. Sub folders under the Jive Group folders that are created on the desktop will not sync to ownCloud or Jive because they will not match the Jive "Group" view. If a sub folder is created under the Jive Group desktop folder, the desktop client will display an error

that this operation is not allowed. For example; if the folder structure is "JiveFileShare/GroupA", any sub folder under GroupA will not be synced to ownCloud or Jive.

Configuring the Jive app

This section explains how each configuration parameter changes the behavior of the app.

Verify https certificate

If your Jive server is under https, it must provide a https certificate when a client connects to it. Curl (the client that ownCloud is using to connect to Jive) usually verify that certificate, but to do that you must somehow supply a CA cert so curl can verify against.

This feature is usually turn off to make the Jive app easier to use, because in this case curl won't verify the certificate, so you don't need to have installed the CA cert. However, turning this off could be a security issue: you could be connecting to a fake Jive server without notice.

If you want to turn on this feature, you must get the CA cert of the server (check "http://curl.haxx.se/docs/sslcerts.html " for more information about how you can get the file you need) and install it in your ownCloud server.

In order to know where you should install the CA cert, you can run

curl -v https://yourserver.com/

You should look the output for a line with the CA path:

- successfully set certificate verify locations:
- CAfile: none
- CApath: /etc/ssl/certs

That's the place where you should install the CA cert.

Once you have installed the CA cert, you should run again the same curl:

curl -v https://yourserver.com/

And look for:

- Server certificate:
- subject: *****
- start date: ******
- expire date: *****
- subjectAltName: *******
- issuer: ******
- SSL certificate verify ok.

If the SSL is verified correctly ("SSL certificate verify ok."), you just need to activate the checkbox.

Curl usually comes installed with some CA certs by default, so all the previous steps might not be needed. Just check that curl can connect to your Jive server, and if so, activate this feature.

Authentication mechanism to use against Jive

To be able to access to Jive, the ownCloud plugin needs to use some kind of authentication. At this time, the plugin supports basic and oAuth authentication.

Basic authentication In order to use basic authentication, you should take into account the following things:

- The credentials used to access to ownCloud must match the ones used to connect to Jive. This means that if you access to ownCloud with a user "PeterP" and password "PeterPassword", the same user must exist in Jive with the same password. Otherwise, the user won't be able to access to Jive.
- If the credentials (typically the password) changes in one side, it must change in the other. You'll need to this manually.

The usage of basic authentication isn't recommended due to the following reasons:

- We need to store the password and be able to recover it. Although the password is stored encrypted, this is not strictly secure.
- Passwords are sent to the server in almost plain text. In fact it's a base64 encoded string of user and password, but that's all the security the authentication provides.

If you plan to use basic authentication, at least make sure you connect through HTTPS protocol and inside a local LAN if possible.

oAuth authentication First of all, make sure Jive is prepared to support this authentication.

The usage of this authentication method solves the issue of having the same credentials in both ownCloud and Jive server. This means that the ownCloud user "PeterP" with password "PeterPassword" can access to the contents of the Jive user "John" with password "John007". It's also possible that another ownCloud user "AliceK" access to the contents of the Jive user "John" too at the same time.

Keep in mind that this isn't insecure: any ownCloud user that wants to access to John's Jive content (following this little example) MUST know his Jive password.

If this authentication method is set, we don't store passwords BUT we still need to store some other things. These things are stored in plain text.

These are the steps to make it work (if your Jive server support this authentication):

- 1. Activate the oAuth authentication in the ownCloud admin settings (just the admin can do this)
- 2. Go to the ownCloud web interface, in the files app. A popup will appear.
- 3. Click on the link that appear in the popup
- 4. You'll get redirected to a Jive page asking for your Jive credentials. If this is not the case, it's recommended to clean the browser cache and start again (to point 2) because you might be accessing to Jive with another user.
- 5. After entering your Jive credentials, you get redirected a page with an activation code. If you entered the wrong credentials, you might not get redirected to that page. If this is the case click in the link again in the ownCloud popup (point 3) which will redirect you to the activation code page.
- 6. Copy the activation code into the ownCloud popup, and click in the "send code" button. If there is no error, you're done.

WARNING:

Not all the oAuth flows are covered by the plugin. The expiration of the access token is handled automatically by the plugin, so it will request a new access token if needed. HOWEVER, the expiration of the refresh token isn't covered, so the plugin will likely stop working for that user (we won't be able to get more access tokens)

[Ask for info to know how to solve this issue?]

It's very important that the user access to ownCloud through the web interface first, so the user goes through the oAuth flow for the first time (as described with the steps above) to get an access token. Otherwise, the plugin won't get an access token and the user won't be able to get the files from Jive.

Jive API URL

You'll need to enter the full URL of the Jive API. This includes the protocol (HTTP or HTTPS) and the port (if any).

An example of API URL could be: "https://myjiveserver.com/api/core/v3/ "

Notice the following things:

- You must specify a protocol that is understandable by curl. Under normal circumstances, the protocol is limited to HTTP or HTTPS.
- If your server is under a port different than the 80, you'll need to specify it. Take " https://jserver.prv:9999/api/core/v3/" as an example
- If your server isn't under the root URL, you can also specify the correct path: "https://myserver.prv:8888/path/to/jive/api/core/v3/"
- The API URL should end with "/api/core/v3/" (be careful with the slash at the end)

Filters

The Jive plugin comes with a set of filters that the admin can set to filter the content the users can access through ownCloud. The drawback of using filters is that there isn't any performance gain because the filtering is mainly done in the ownCloud side, and even can degrade performance in some cases. We'll explain the filters one by one, and tell you what consequences have each one.

Category filter and separator You can filter files using one or several categories. This filter applies only to groups and files inside those groups. Your private files won't be affected by this filter.

In order to set this filter, you can provide a list of categories, all in one line. In order to separate the different categories, you must use the separator set in the "category separator" text box.

Jive category filter : syncWithMe,sync,syncMe

Jive category separator : ,

You can also achieve the same behavior with:

Jive category filter : syncWithMe#sync#syncMe

Jive category separator : #

The plugin will show all groups which have ALL those categories set. If there is a group with any of the categories missing, that group won't be shown. Anyway, you should only need to set one category.

It's important to notice that, although you can set only one category or leave the text box empty, the category separator MUST always be set. In fact, you shouldn't need to change the default value of the category separator.

Files shown inside those groups will be also affected by this filter. This means that all the files shown inside those groups must have all the categories too.

Files uploaded through ownCloud to those groups will have all the categories set in Jive automatically. If you want to add more categories to those files, you'll need to do it manually through Jive.

The usage of the category filter can degrade the performance a lot. We need to make extra calls to Jive to get the categories for each group, one extra call per group returned by Jive in the first place. There is also the limitation of not having more than 25 categories set per group. Use this filter with extreme caution.

You can "disable" this filter just by setting the category filter empty. This will prevent the extra call from being made, and will show all available groups.

Tag filter This filter works only for private files. Files inside groups won't be affected by this filter.

You can only set one tag for the files that will be shown in ownCloud. Make sure you set one of the tags from Jive as they're shown there. It's highly recommended to use only lowercase letters to set the tag to prevent possible issues when the tag is set in Jive.

The usage of this filter won't alter significantly the performance

It's important to notice that the filter will be applied to all users. Users won't be able to set their own tag to sync their own files.

This filter can also be "disabled" by setting the filter empty.

Forbidden extensions filter and separator This filter is set the same way as the category filter: you provide a list of extensions that are separated by the char set in the separator text box.

Jive forbidden extensions: .exe,.zip,.tar.gz

Jive forbidden extensions separator : ,

You can also achieve the same behavior with:

Jive forbidden extensions: .exe#.zip#.tar.gz

Jive forbidden extensions separator: #

Keep in mind that the filter is performed against the end of the filename, that's why the "." is important. If you set "exe" as a forbidden extension, a file named "texe" or "f1.lexe" will be affected by this filter.

You must also take into account that, by using only the filename, we avoid to download the file, so the performance isn't significantly degraded. On the other hand, we cannot verify that a ".png" file is what it claims to be.

This filter works for any file, and for uploads and downloads through ownCloud. This means that you won't be able to upload a file with any of those extensions from onwCloud and the Jive files which have those extensions won't be shown (and consequently they won't be downloaded). Of course, you can still upload the files from Jive (if Jive allows it) and have them there.

Maximum upload file size This filter allows you to limit the size of the files that will go through ownCloud. All files uploads and downloads will be affected by this filter. You won't be able to upload files bigger than the file size limit and the Jive files bigger than the limit won't be shown in ownCloud (and consequently they won't be downloaded)

Under normal circumstances, you want to match the limit with the one Jive has. This way you can notify errors regarding the file size faster because the files won't reach the Jive server, and at the same time you allow the users to upload up to the maximum limit that Jive allows. (Note: we can't know this limit from ownCloud, so we can't provide a sensitive default value, plus the value can change among Jive instances. You might need to adjust the value manually).

You can also set the limit to a lower value than what it's in Jive, so only small files will be delivered from ownCloud.
Show groups of which you are member Under normal circumstances, you can see all available groups in Jive, including open, member-only and private groups, only secrets groups are outside. Even if you're not a member of those groups, you can still see their contents.

For small Jive installations (less than 100 available groups per user) this is usually enough, and it has an acceptable performance. However, for larger installations, with more than 500 groups available per user, the performance is degraded a lot.

For these larger installations, this checkbox comes in handy.

Again, under normal circumstances, it's common that a user is member of just a few groups (let's say less than 25) even if there are thousand of groups available that the user can see. It usually makes sense to show the contents of only those 25 groups, not every group available.

By activating this checkbox, the user will see only those 25 groups instead of all available groups. This will increase the performance a lot, specially for larger installations, as long as each user isn't member of too many groups. Anyway, if there are user who are member of too many groups, the performance will still be degraded.

FS mount points

This Jive plugin mounts one (or two) virtual filesystems on the normal one in a transparent way.

From a user point of view, these virtual filesystems appear as new folders inside the root one.

From the settings page, you can change the mount points names. The folders will change accordingly.

Jive FS mount point The name of the folder that will hold the Jive virtual FS. The name shouldn't collide with any existing name in the root folder to prevent possible issues. The virtual FS will be mounted inside the root folder of the ownCloud FS.

As said, the contents of the folder will be the groups that the user can access through ownCloud (recheck the "filters" section).

Jive private folder The folder where your private Jive files will be stored. The name of the folder will be the same for all users, although the contents will likely differ.

This private folder will be inside the Jive mount point, as if it were another group.

Files inside this folder will be only visible to you, but they will be stored in Jive. They won't be visible neither for ownCloud users nor Jive users.

In order to prevent collisions with other groups, the folder name might be changed automatically by adding "(private)" to the end of the folder name if it's needed .

Large file sharing subsystem The large file sharing allow you to share files over the Jive limits (typically size limits). You can enable or disable this subsystem by checking or un-checking the checkbox, and provide the corresponding mount point. Use a non-existent folder name to prevent issues.

Files inside that folder will be stored inside the ownCloud server. However those files can be shared by link to Jive.

The process is like the following:

- 1. Upload a file (or folder) inside the large file sharing folder (by default named as "Too Big For")
- 2. Once the file is uploaded, click in the "share" action, and then click in the "Share link" checkbox
- 3. By default the share link will expire after 1 week. You can change the value and / or protect the link by password
- 4. Click the "Submit to Jive" button (the name can be changed depending on the actual Jive folder name)

5. A new browser tab should appear with the Jive draft ready to be edited (you might need to enter your Jive credentials first). The draft will have some predefined text, but you can edit it to your needs. Once you publish the document, it's done.

Notifications

This Jive plugin runs a connectivity check between ownCloud and Jive whenever the web page is loaded. This check allows you to know some potential issues between the ownCloud – Jive connection.

When a potential issue is detected, a notification will be shown, so you'll know what's happening.

You can control the time the notification is shown in the "notification time for the connectivity check" configuration. The time is in seconds.

12.5 User Management (Enterprise only)

12.5.1 Shibboleth Integration (Enterprise only)

Introduction

The ownCloud Shibboleth user backend application integrates ownCloud with a Shibboleth Service Provider (SP) and allows operations in federated and single-sign-on (SSO) infrastructures. Setting up Shibboleth has two big steps:

- 1. Enable and configure the Apache Shibboleth module.
- 2. Enable and configure the ownCloud Shibboleth app.

The Apache Shibboleth module

Currently supported installations are based on the native Apache integration. The individual configuration of the service provider is highly dependent on the operating system, as well as on the integration with the Identity Providers (IdP), and require case-by-case analysis and installation.

A good starting point for the service provider installation can be found in the official Shibboleth Wiki.

A successful installation and configuration will populate Apache environment variables with at least a unique user id which is then used by the ownCloud Shibboleth app to login a user.

See the documentation Wiki for more configuration examples.

Apache Configuration

This is an example configuration as installed and operated on a Linux server running the Apache 2.4 Web server. These configurations are highly operating system specific and require a high degree of customization.

The ownCloud instance itself is installed in /var/www/owncloud/. The following aliases are defined in an Apache virtual host directive:

```
# non-Shibboleth access
Alias /owncloud /var/www/owncloud/
# for Shibboleth access
Alias /oc-shib /var/www/owncloud/
```

Further Shibboleth specific configuration as defined in /etc/apache2/conf.d/shib.conf:

```
#
# Load the Shibboleth module.
#
LoadModule mod_shib /usr/lib64/shibboleth/mod_shib_24.so
#
# Ensures handler will be accessible.
#
<Location /Shibboleth.sso>
 AuthType None
 Require all granted
</Location>
#
# Configure the module for content.
#
#
# Besides the exceptions below, this location is now under control of
# Shibboleth
#
<Location /oc-shib>
 AuthType shibboleth
 ShibRequireSession On
 ShibUseHeaders Off
 ShibExportAssertion On
 require valid-user
</Location>
#
# Shibboleth is disabled for the following location to allow non
# shibboleth webdav access
<Location ~ "/oc-shib/remote.php/nonshib-webdav">
 AuthType None
 Require all granted
</Location>
#
# Shibboleth is disabled for the following location to allow public link
# sharing
#
<Location ~ \
"/oc-shib/(status.php$\
lindex.php/s/\
|public.php\
|cron.php$\
|core/img/\
lindex.php/apps/files_sharing/ajax/publicpreview.php$\
|index.php/apps/files/ajax/upload.php$\
|apps/files/templates/fileexists.html$\
lindex.php/apps/files/ajax/mimeicon.php$\
|index.php/apps/files_sharing/ajax/list.php$\
|themes/\
lindex.php/apps/files_pdfviewer/\
|apps/files_pdfviewer/)">
 AuthType None
  Require all granted
```

```
</Location>
#
# Shibboleth is disabled for the following location to allow public gallery
# sharing
#
<Location ~ \
"/oc-shib/(index.php/apps/gallery/s/
|index.php/apps/gallery/slideshow$\
|index.php/apps/gallery/.*\.public)">
 AuthType None
 Require all granted
</Location>
#
# Shibboleth is disabled for the following location to allow public link
# sharing
#
<Location ~ "/oc-shib/.*\.css">
  AuthType None
  Require all granted
</Location>
#
# Shibboleth is disabled for the following location to allow public link
# sharing
#
<Location ~ "/oc-shib/.*\.js">
 AuthType None
 Require all granted
</Location>
#
# Shibboleth is disabled for the following location to allow public link
# sharing
<Location ~ "/oc-shib/.*\.woff">
 AuthType None
  Require all granted
</Location>
```

Depending on the ownCloud Shibboleth app mode, you may need to revisit this configuration.

The ownCloud Shibboleth App

After enabling the Shibboleth app on your Apps page, you need to choose the app mode and map the necessary Shibboleth environment variables to ownCloud user attributes on your Admin page.

Choosing the App Mode

After enabling the app it will be in **Not active** mode, which ignores a Shibboleth session and allows you to login as an administrator and inspect the currently available Apache environment variables. Use this mode to set up the environment mapping for the other modes, and in case you locked yourself out of the system. You can also change the app mode and environment mappings by using the occ command, like this example on Ubuntu Linux:

Shibboleth		
App Mode Not active	•	
Not active Environme Autoprovision U	lsers	
Use Shib-Session-ID	only •	as Shibboleth session
Use eppn	•	as uid
Use eppn	•	as email
Use eppn	•	as display name
Server Environment:		
htaccessWorking HTTP_HOST HTTP_USER_AGENT HTTP_ACCEPT HTTP_ACCEPT_LANGUAGE HTTP_ACCEPT_ENCODING HTTP_DNT	true docker.oc Mozilla/5. text/html, en-US,en; gzip, defla 1	.solidgear.es:53738 0 (X11; Ubuntu; Linux x86_64; rv:42.0) G application/xhtml+xml,application/xml; q=0.5 ate
HTTP_COOKIE HTTP_CONNECTION PATH	PHPSESSII keep-alive /usr/local/	D=nlkrv949lmuo7dpkkgb91gbe13; ocy0: e /sbin:/usr/local/bin:/usr/sbin:/usr/bin:/!
SERVER_SIGNATURE SERVER_SOFTWARE SERVER NAME	<address Apache/2 docker.oc</address 	>Apache/2.4.7 (Ubuntu) Server at docke .4.7 (Ubuntu) .solidgear.es
SERVER_ADDR SERVER_PORT REMOTE_ADDR	172.17.1.2 53738 166.176.1	85.154
DOCUMENT_ROOT REQUEST_SCHEME	/opt/owno http	loud

Figure 12.4: figure 1: Enabling Shibboleth on the ownCloud Admin page

\$ sudo -u www-data php occ shibboleth:mode notactive \$ sudo -u www-data php occ shibboleth:mapping --uid login

In **Single sign-on only** mode the app checks if the environment variable for the Shibboleth session, by default **Shib-Session-Id**, is set. If that is the case it will take the value of the environment variable as the uid, by default eppn, and check if a user is known by that uid. In effect, this allows another user backend, e.g., the LDAP app, to provide the displayname, email and avatar.

Note: As an example the IdP can send the **sAMAccountName** which the Apache Shibboleth module writes to a custom Apache environment variable called login. The ownCloud Shibboleth app reads that login environment variable and tries to find an LDAP user with that uid. For this to work the LDAP backend also needs to be configured to use the **sAMAccountName** as the **Internal Username Attribute** in the *LDAP expert settings*.

Note: In many scenarios Shibboleth is not intended to hide the user's password from the service provider, but only to implement SSO. If that is the case it is sufficient to protect the ownCloud base url with Shibboleth. This will send Web users to the IdP but allow desktop and mobile clients to continue using username and password, preventing popups due to an expired Shibboleth session lifetime.

In **Autoprovision Users** mode the app will not ask another user backend, but instead provision users on the fly by reading the two additional environment variables for display name and email address.



Figure 12.5: figure 2: Mapping Shibboleth environment configuration variables to ownCloud user attributes

In ownCloud 8.1 the Shibboleth environment variable mapping was stored in apps/user_shibboleth/config.php. This file was overwritten on upgrades, preventing a seamless upgrade procedure. In ownCloud 8.2+ the variables are stored in the ownCloud database, making Shibboleth automatically upgradeable.

Shibboleth with Desktop and Mobile Clients

The ownCloud Desktop Client can interact with an ownCloud instance running inside a Shibboleth Service Provider by using built-in browser components for authentication against the IdP.

The regular ownCloud Android and iOS mobile apps do not work with Shibboleth. However, customers who create *branded mobile apps with ownBrander* have the option to enable SAML authentication in ownBrander.

Enterprise customers also have the option to request a regular ownCloud mobile client built to use Shibboleth from their ownCloud account representatives.

The ownCloud desktop sync client and mobile apps store users' logins, so your users only need to enter their logins the first time they set up their accounts.

Note: The ownCloud clients may use only a single Shibboleth login per ownCloud server; multi-account is not supported with Shibboleth.

These screenshots show what the user sees at account setup. Figure 1 shows a test Shibboleth login screen from Testshib.org on the ownCloud desktop sync client.

ownCloud - TestShib Identity Provider Login (idp.testshib.org)	×
TESTSHIB	Â
TestShib Identity Provider Login	
Authenticating to testshib.example.com	Ľ
Username:	
Password:	
Login	
	\sim

Figure 12.6: figure 3: First login screen

Then after going through the setup wizard, the desktop sync client displays the server and login information just like it does for any other ownCloud server connections.

To your users, it doesn't look or behave differently on the desktop sync client, Android app, or iOS app from an ordinary ownCloud account setup. The only difference is the initial setup screen where they enter their account login.



Figure 12.7: figure 4: ownCloud client displays server information

WebDAV Support

Users of standard WebDAV clients can use an alternative WebDAV URL, for example https://cloud.example.com/remote.php/nonshib-webdav/ to log in with their username and password. The password is generated on the Personal settings page.

Non Shibboleth WebDAV

To access your files through WebDAV, please use the following URL:

https://www.www.www.www.remote.php/nonshib-webdav/ Credentials

For WebDAV, you must use separate credentials. Please use the following:

Username: myself@testshib.org

Password:

Generate password Note: after generating, the password is visible only oncel

Note: In **Single sign-on only** mode the alternative WebDAV Url feature will not work, as we have no way to store the WebDAV password. Instead the normal WebDAV endpoint can be omitted from the Shibboleth authentication, allowing WebDAV clients to use normal username and password based authentication. That includes the desktop and mobile clients.

For provisioning purpose an OCS API has been added to revoke a generated password for a user:

Syntax: /v1/cloud/users/{userid}/non_shib_password

```
• HTTP method: DELETE
```

Status codes:

- 100 successful
- 998 user unknown

Example:

```
$ curl -X DELETE "https://cloud.example.com/ocs/v1.php/cloud/users/myself@testshib.org/non_shib_pass
<?xml version="1.0"?>
<ocs>
<meta>
  <status>ok</status>
   <status>ok</status>
   <statuscode>100</statuscode>
   <message/>
   </meta>
   <data/>
</ocs>
```

Known Limitations

Encryption

File encryption can only be used together with Shibboleth when the *master key-based encryption* is used because the per- user encryption requires the user's password to unlock the private encryption key. Due to the nature of Shibboleth

the user's password is not known to the service provider.

Other Login Mechanisms

You can allow other login mechanisms (e.g. LDAP or ownCloud native) by creating a second Apache virtual host configuration. This second location is not protected by Shibboleth, and you can use your other ownCloud login mechanisms.

Session Timeout

Session timeout on Shibboleth is controlled by the IdP. It is not possible to have a session length longer than the length controlled by the IdP. In extreme cases this could result in re-login on mobile clients and desktop clients every hour.

UID Considerations and Windows Network Drive compatability

When using user_shibboleth in Single sign-on only mode, together with user_ldap, both apps need to resolve to the same uid. user_shibboleth will do the authentication, and user_ldap will provide user details such as email and displayname. In the case of Active Directory, multiple attributes can be used as the uid. But they all have different implications to take into account:

sAMAccountName

- Example: jfd
- Uniqueness: Domain local, might change e.g. marriage
- Other implications: Works with windows_network_drive app

userPrincipalName

- *Example:* jfd@owncloud.com
- Uniqueness: Forest local, might change on eg. marriage
- Other implications: TODO check WND compatability

objectSid

- *Example:* S-1-5-21-2611707862-2219215769-354220275-1137
- Uniqueness: Domain local, changes when the user is moved to a new domain
- Other implications: Incompatible with windows_network_drive app

sIDHistory

- Example: Multi-value
- Uniqueness: Contains previous objectSIDs
- Other implications: Incompatible with windows_network_drive app

objectGUID

- *Example:* 47AB881D-0655-414D-982F-02998C905A28
- Uniqueness: Globally unique
- Other implications: Incompatible with windows_network_drive app

Keep in mind that ownCloud will derive the home folder from the uid, unless a home folder naming rule is in place. The only truly stable attribute is the <code>objectGUID</code>, so that should be used. If not for the uid then at least as the home folder naming rule. The trade off here is that if you want to use <code>windows_network_drive</code> you are bound to the <code>sAMAccountName</code>, as that is used as the login.

Also be aware that using user_shibboleth in Autoprovision Users mode will not allow you to use SSO for additional user_ldap users, because uid collisions will be detected by user_ldap.

12.6 Enterprise File Management (Enterprise Only)

12.6.1 Enabling Anonymous Uploads with Files Drop (Enterprise Only)

The Files Drop application, introduced in ownCloud 8.0.3 Enterprise Subscription, allows anyone to upload files with the click of a button to the directory of your choosing, without needing a login, and they cannot see or change the contents of the directory. It is the perfect replacement for attaching large files to email, maintaining an FTP server, and commercial file-sharing services.

When files are uploaded to your Files Drop directory, you can manage them just like any other ownCloud share: you may share them, restrict access, edit, and delete them.

Setting Up the Files Drop App

Setting up Files Drop is a matter of a few clicks. First go to your Apps page and enable it.



Files Drop 0.4.1

by ownCloud Inc. / Tom Needham and Thomas Müller (Commerciallicensed)

✓ Official

Creates a link for anonymous upload into a directory of your choice, which may then be shared normally within ownCloud.

Hide description ...

Disable

Now your users will see a configuration section on their Personal pages.

Files Drop

Choose the folder to use for anonymous upload

Choose

Click the **Choose** button to open a dialog to select your upload directory. You may wish to first create a special upload directory (on your Files page), which in the following example is name **upload**.

Choose the folder to store files uploaded with File Drop	×
#	
Documents	an hour ago
Photos	an hour ago
uploads	an hour ago
ownCloudServerAdminManual.pdf	an hour ago
ownCloudUserManual.pdf	an hour ago
	Choose

On your Personal page you should now see a URL for your upload directory. Share this URL with anyone you want to allow uploads to your File Drop folder. Note that the maximum upload size in this example is 512MB. (The default ownCloud upload file size limit is 512MB. See *Uploading big files* > 512MB to learn how to customize this.)

Files Drop

Folder used for your anonymous upload endpoint: /uploads

http://ubuntu-server/owncloud/index.php/apps/files_drop/lmeutxejsywi

Free space: 4 GB - Max file upload size: 512 MB

Using the Files Drop App

Uploading files via the Files Drop app is simple. Open your Web browser to the share URL created by ownCloud:

Click the **Click to upload file** button. This opens a file picker, and you select the file or directory you want to upload.

When your upload is completed, you'll see a confirmation message with the filenames.

12.6.2 Advanced File Tagging With the Workflow App (Enterprise only)

New in ownCloud 9.0, the Workflow App provides advanced management of file tagging. The app has three parts: Tag Manager, Automatic Tagging, and Retention.

The Workflow App should be enabled by default (Apps page), and the three configuration modules visible on your ownCloud Admin page.

See Tagging Files in the ownCloud User manual to learn how to apply and filter tags on files.



(Select File		-	• ×
📝 🔇 🛧 carla				
<u>P</u> laces	Name	 Size 	Modified	
Search	🔳 Desktop		07/24/2014	
Recently Used	🚚 Downloads		14:44	
🏠 carla	🚚 ownCloud		11/05/2014	
Desktop	🚚 Templates		07/24/2014	
File System	oc8-packages.txt	1.8 kB	02/23/2015	
SVBOXADDITION	oc8-server-packages.txt	1.9 kB	02/23/2015	
	Release.key	4.5 kB	03/18/2015	
	Release.key.1	4.5 kB	03/30/2015	
:	server-and-enterprise-installation	4.7 kB	03/18/2015	
÷ -				=
		🔀 <u>C</u> ance	el 🧾 <u>O</u> per	n



Tag Manager

The Tag Manager is for creating new tags, editing existing tags, and deleting tags. Tags may be made **Visible and** assignable by all users, Not-assignable by normal users, or Invisible to normal users.

Visible and assignable all users means that users may see, rename. and apply admin-created tags to files and folders.

Not-assignable by normal users means tags are read-only, and users cannot assign them to files or folders.

Invisible to normal users means visible only to ownCloud admins.

Workflow
Tag manager
Edit an existing tag
Add new tag
season schedule
Visible and assignable by all users -
Visible and assignable by all users
Not-assignable by normal users Invisible to normal users

Automatic Tagging

The Automatic Tagging module operates on newly-uploaded files. Create a set of conditions, and then when a file or folder matches those conditions it is automatically tagged. The tag must already have been created with the Tag Manager.

For example, you can assign the invisible tag **iOS Uploads** to all files uploaded from iOS devices. This tag is visible only to admins.

When files with this tag are shared with you, you can view them with the Tags filter on the Files page.

Automatic Tagging is especially useful with the Retention module.

Retention

The Retention module is your housecleaning power tool, because it automatically deletes files after a time period that you specify. Select which tag to set a time limit on, and then set your time limit. File age is calculated from the file mtime (modification time).

For best performance, retention tags should be applied high in your file hierarchy. If subfolders have the same tags as their parent folders, their tags must also be processed, so it will take a little longer.

Automatic tagging

Automatically tag newly uploaded files, matching the conditions, with the following tags:

000	C11	-3	-
105	tiles		
100	11100	(F	

Conditions: Device type is iOS Client Add tags: iOS Uploads (invisible)

+ Add new rule

Ŕ	ownCloud - Enterpr	Ise
	All files	* iOS Uploads (invisible)
*	Favorites	Name 🔺
<	Shared with you	Photo-2016-03-18-11-17-04_0014.PNG
<	Shared with others	1 file
S	Shared by link	
٩	Tags	
	Retention periods	
	Delete files tagged with the fo	llowing tags after the given time:
	iOS Uploads (invisible)	24 Days 🔻 👕
	+ Add new rule	

Retention Engines

There are two retention engines that further allow you to fine-tune your retention settings: **TagBasedRetention** and **UserBasedRetention**. **TagBasedRetention** is the default.

TagBasedRetention: This checks files that have a particular tag assigned. Then it checks (depth-first) the children of the tagged item, before continuing with the other tagged items. Children that have already been checked will not be checked a second time.

This is optimised for processing smaller numbers of files that have multiple retention tags.

UserBasedRetention: Examines files per user. It first iterates over all files and folders (siblings first), then examines the tags for those items and checks their respective retention periods. This is optimised for many files with few retention tags.

To select UserBasedRetention, add this line to your ee.config.php:

'workflow.retention_engine' => userbased,

12.7 Enterprise Logging Apps (Enterprise only)

12.7.1 Enterprise Logging Apps

The Log user and file sharing actions app (apps/admin_audit) records the file sharing activity of your users, file tagging, and user logins and logouts.



Your logging level must be set to at least Info, warnings, errors, and fatal issues on your ownCloud admin page, or 'loglevel' => 1 in config.php.

View your logfiles on your admin page. Click the **Download logfile** button to dump the plain text log, or open the logfile directly in a text editor. The default location is owncloud/data/owncloud.log.

See Logging Configuration and Advanced File Tagging With the Workflow App (Enterprise only) for more information on logging and tagging.

12.8 Enterprise Firewall (Enterprise only)

12.8.1 File Firewall (Enterprise only)

The File Firewall GUI enables you to create and manage firewall rule sets from your ownCloud admin page. The File Firewall gives you finer-grained control of access and sharing, with rules for allowing or denying access, and restrictions per group, upload size, client devices, IP address, time of day, and many more criteria. For additional flexibility the File Firewall also supports regular expressions.

Each rule consists of one or more conditions. A request matches a rule if all conditions evaluate to true. If a request matches at least one of the defined rules, the request is blocked and the file content can not be read or written.

Note: As of ownCloud 9.0, the File Firewall app cannot lock out administrators from the Web interface when rules are misconfigured.

Figure 1 shows an empty firewall configuration panel. Set your logging level to **Failures Only** for debugging, and create a new ruleset by clicking the **Add Group** button. After setting up your rules you must click the **Save Rules** button.

File Firewall

Save Rules

Requests are checked against all rules that are defined below. A request is blocked, when at least one rule matches the request. A rule matches a request, when all conditions evaluate to true.

Group Name			
•			x Delete
		+ Add rule	+ Add group
Logging			

Figure 12.8: Figure 1: Empty File Firewall configuration panel

Figure 2 shows two rules. The first rule, **No Support outside office hours**, prevents members of the support group from logging into the ownCloud Web interface from 5pm-9am, and also blocks client syncing.

The second rule prevents members of the qa-team group from accessing the Web UI from IP addresses that are outside of the local network.

All other users are not affected, and can log in anytime from anywhere.

Available Conditions

User Group The user (islis not) a member of the selected group.

User Agent The User-Agent of the request (matchesldoes not match) the given string.

Failures Only

File Firewall

Requests are checked against all rules that are defined below. A request is blocked, when at least one rule matches the request. A rule matches a request, when all conditions evaluate to true.

No Support outside						
User Group 🔹	is 🔻	support 👻				x Delete
Request Time 🔹	between 🖣	05:00 pm -0700	, 09:00 am	-0700		X Delete
					+ Add rule	+ Add group
No QA outside of th						
User Group 🔹	is 💌	qa-team 👻				X Delete
Request IP Range 🔹	is 🔻	192.168.1.10/24				X Delete
			•	Add rule	+ Add grou	p X Delete
Logging						
Save Rules Failures On	ly 🔻 Firew	all rules saved.				

Figure 12.9: Figure 2: Two example rules that restrict logins per user group

User Device A shortcut for matching all known (android lios | desktop) sync clients by their User Agent string.

Request Time The time of the request (has tolmust not) be in a single range from beginning time to end time.

Request URL The full page URL (has tolmust not) (matchlcontainlbegin withlend) with a given string.

Request Type The request (islis not) a (WebDAV|public share linklother) request.

- Request IP Range (IPv4) and IP Range (IPv6) The request's REMOTE_ADDR header (islis not) matching the given IP range.
- Subnet (IPv4) and Subnet (IPv6) The request's SERVER_ADDR header (islis not) matching the given IP range.
- File Size Upload When a file is uploaded the size has to be (lesslless or equallgreater/greater or equal) to the given size.
- File Mimetype Upload When a file is uploaded the mimetype (islis notlbegins withldoes not begin withlends withldoes not end with) the given string.

System File Tag One of the parent folders or the file itself (islis not) tagged with a System tag.

- **Regular Expression** The File Firewall supports regular expressions, allowing you to create custom rules using the following conditions:
 - IP Range (IPv4)
 - IP Range (IPv6)
 - Subnet (IPv4)
 - Subnet (IPv6)
 - User agent
 - User group
 - Request URL

You can combine multiple rules into one rule. E.g., if a rule applies to both the support and the qa-team you could write your rule like this:

Regular Expression > ^(support|qa-team)\$ > is > User group

No Manual Editing

We do not recommend modifying the configuration values directly in your config.php. These use JSON encoding, so the values are difficult to read and a single typo will break all of your rules.

Controlling Access to Folders

The easiest way to block access to a folder, starting with ownCloud 9.0, is to use a system tag. A new rule type was added which allows you to block access to files and folders, where at least one of the parents has a given tag. Now you just need to add the tag to the folder or file, and then block the tag with the File Firewall.

This example blocks access to any folder with the tag "Confidential".

Block by System Tag:

System file tag: is "Confidential" Subnet IPv4: is not "255.255.255.0/24"

File Firewall

Requests are checked against all rules that are defined below.

Block confidential files		
System file tag 🔹	is 🔻	Confidential (visible, not assignable) v
Subnet (IPv4) •	is not 🔻	255.255.255.0/24

Custom Configuration for Branded Clients

If you are using *branded ownCloud clients*, you may define firewall.branded_clients in your config.php to identify your branded clients in the firewall "User Device" rule.

The configuration is a User-Agent => Device map. Device must be one of the following:

- android
- android_branded
- ios
- ios_branded
- desktop
- desktop_branded

The User-Agent is always compared all lowercase. By default the agent is compared with equals. When a trailing or leading asterisk, *, is found, the agent is compared with starts with or ends with. If the agent has both a leading and a trailing *, the string must appear anywhere. For technical reasons the User-Agent string must be at least 4 characters (including wildcards). (When you build your branded client you have the option to create a custom User Agent.)

In this example configuration you need to replace the example User Agent strings, for example 'android_branded', with your own User Agent strings:

// config.php

```
'firewall.branded_clients' => array(
    'my ownbrander android user agent string' => 'android_branded',
    'my ownbrander second android user agent string' => 'android_branded',
    'my ownbrander ios user agent string' => 'ios_branded',
    'my ownbrander second ios user agent string' => 'ios_branded',
    'my ownbrander desktop user agent string' => 'desktop_branded',
    'my ownbrander second desktop user agent string' => 'desktop_branded',
    'my ownbrander second desktop user agent string' => 'desktop_branded',
    'my ownbrander second desktop user agent string' => 'desktop_branded',
    'my ownbrander second desktop user agent string' => 'desktop_branded',
    'my ownbrander second desktop user agent string' => 'desktop_branded',
```

The Web UI dropdown then expands to the following options:

• Android Client - always visible

- iOS Client always visible
- Desktop Client always visible
- Android Client (Branded) visible when at least one android_branded is defined
- iOS Client (Branded) visible when at least one ios_branded is defined
- Desktop Client (Branded) visible when at least one desktop_branded is defined
- All branded clients visible when at least one of android_branded, ios_branded or desktop_branded is defined
- All non-branded clients visible when at least one of android_branded, ios_branded or desktop_branded is defined
- Others (Browsers, etc.) always visible

Then these options operate this way:

- The * Client options only match android, ios and desktop respectively.
- The * Client (Branded) options match the *_branded agents equivalent.
- All branded clients matches: android_branded, ios_branded and desktop_branded
- All non-branded clients matches: android, ios and desktop

12.9 Enterprise Troubleshooting

When you have problems with your ownCloud Enterprise installation, refer to *General Troubleshooting* to see if you can resolve your issue without opening a support ticket. If you need to open a support ticket, use the Open Ticket button in your account on https://customer.owncloud.com/owncloud/.

Bug reports and trouble tickets usually need a copy of your ownCloud server configuration report. You have two ways to generate a configuration report.

- 1. Use the occ config command.
- 2. Use the **Enterprise license key** app on your ownCloud Admin page to generate the report with the click of a button.

Enterprise license key

The registered enterprise license key expires in 29 days.

Download ownCloud config report

Both methods automatically obscure passwords and secrets.