ownCloud Architectural Guide

Performance and scaling of ownCloud Enterprise

ownCloud is the open platform for more productivity and security in digital collaboration and provides a common file access layer regardless of where the data lives – in applications, object stores, on-premise storage or in the cloud. Data is kept where it is while IT can manage proprietary information and business risk; leveraging existing data management, security and governance tools and processes. Whether in SharePoint, on a Windows network drive or in cloud storage, users have a single interface from which they can access, sync and share files on any device, anytime, from anywhere – all completely managed, secured and controlled by IT.

This paper describes the performance and scaling of the industry-standard EFSS (Enterprise File Sync Share) solution ownCloud Enterprise. ownCloud Enterprise runs on an Enterprise LAMP (Linux Apache MySQL PHP) stack on either Ubuntu, Debian, Red Hat, CentOS, SUSE operating systems running PHP connecting to one of the supported databases, MySQL, PostgreSQL, MariaDB or Oracle DB.

Intended Audience

This paper is intended for System Architects and Administrators who are deploying ownCloud Enterprise into their IT infrastructure. It is assumed that the reader has a basic understanding of IT infrastructure, basic networking and routing skills, along with virtualization and server installation best practices. It is also assumed that the reader has a basic understanding of the principles of ownCloud and its capabilities and has had at least limited hands-on experience with ownCloud Enterprise's user-interface as well as administrator interface.

With over 50 million community and enterprise users, ownCloud is the preferred file access solution for organizations across the globe.

Solution Architecture Overview

The core of the ownCloud solution is the ownCloud server. Unlike consumer-grade file-sharing services, ownCloud's Enterprise solution enables IT to protect and manage files within the ownCloud environment – from file storage to user / group provisioning. ownCloud monitors and logs all data access events for downstream auditing and analysis using popular tools like Splunk®.

The server provides a secure web interface through which administrators control all of ownCloud's resources, allowing authorized users to enable and disable features, set policies, create shares and manage users. Advanced features for enterprise directory integration and file-firewalls give admins exceptional flexibility and control. The server also manages and secures API access to ownCloud, while providing the internal processing engine needed to deliver high-performance file-sharing services. In addition to that, ownCloud also provides an automated way to attach tags to files according to a set of rules. These tags can also be used within the file firewall to control the access and sharing of files.

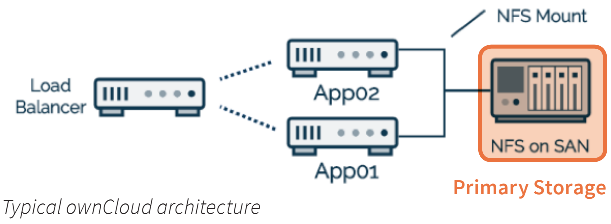
To enable a broad range of storage alternatives, ownCloud also abstracts the storage tier. As a result, ownCloud can leverage just about any storage protocol that can be mounted on your ownCloud server at the operating system level as **Primary Storage**. ownCloud also has native connectors for object stores here, allowing us to directly access objects stores like S3. Other storage resources can also be mounted on the system using ownCloud Enterprise's internal connectors, otherwise known as **Secondary Storage**.

Primary

Local Partition, S3, NFS, GFS, GFS2, XFS, Red Hat Storage, ZFS GPFS, etc.

Primary Storage

Primary Storage is required in all ownCloud installations. By default, ownCloud utilizes the `../ownCloud/` data directory of the root of the webserver on the local partition as the default user data directory. It‘s presence is required by the ownCloud core application to store user-specific metadata such as thumbnails, temporary files, cache, encryption keys, versioning and trash-bin. While the ownCloud data directory can reside on the local partition of the underlying server, administrators can elect to utilize other means of Primary Storage attached / mounted to the local Linux operating system.



Typical ownCloud architecture

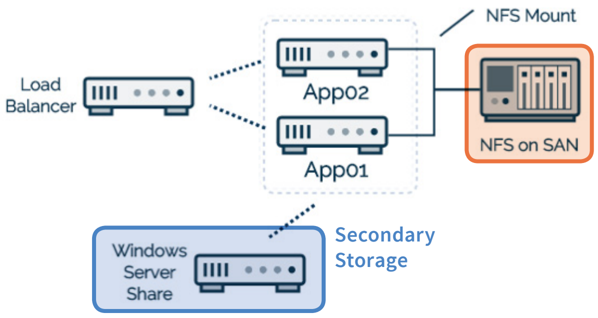
ownCloud can utilize any files system that the local operating system can mount, such as SMB, NFS, GFS, etc , and has native connectors for object stores. Once an administrator elects to utilize a storage mechanism other than the local partition, the ownCloud configuration file simply needs to be pointed to the new storage location. While ownCloud can seamlessly work with the Primary Storage on the local partition, administrators will want to utilize an external Primary Storage source mounted to the Linux operating system to ease in storage overlap, performance, backups / snapshots and expandability concerns. Use of object stores as primary storage is done through a specific ownCloud plugin. For larger installs of ownCloud where multiple instances of the ownCloud Enterprise application servers are running behind a load balancer, external Primary Storage is a requirement as it allows access to the user storage from multiple application servers that reside in the web-farm or cluster.

Secondary

CIFS, WebDAV, FTP, SFTP, DropBox, Windows Share, Google, ownCloud, etc.

Secondary Storage

Secondary Storage is optional, but particularly robust component of ownCloud Enterprise that will allow organizations to utilize their existing data infrastructure. This not only facilitates flexibility in design, but it also allows administrators to utilize existing data silos and ACL (Access Control Lists), minimizing the need for redundant storage devices, as ownCloud Enterprise supports Amazon S3, Dropbox, Google Drive, FTP, SFTP, SMB / CIFS, SharePoint, Windows server shares and WebDAV.



Secondary Storage architecture with ownCloud

The diversity of the various compatible Secondary Storage along with the Primary Storage aspect of ownCloud Enterprise gives businesses an easily customizable solution that can integrate data from a number of sources. Think of ownCloud Enterprise as a switch or router for your current existing data silos, allowing seamless integration and presenting the end-user with an intuitive yet simple interface to multiple data sources. For example, administrators may want to utilize existing Windows Server network shares of a department already in place, minimizing data redundancy and preserving ACLs. In this case, ownCloud Enterprise, through the use of the Windows Network Drive (WND), could connect to an existing Windows Share `\\servername\engineering` and allow users to securely access data while at work, home or on the road and collaborate with internal colleagues or remotely with contractors.

As with all Secondary Storage within ownCloud, such connections, once configured, are stored in the ownCloud Enterprise database. This helps facilitate horizontal scaling by avoiding numerous configuration steps as well any changes or additions to Secondary Storage is effective immediately and available across an ownCloud Enterprise cluster.

Horizontal versus Vertical sizing

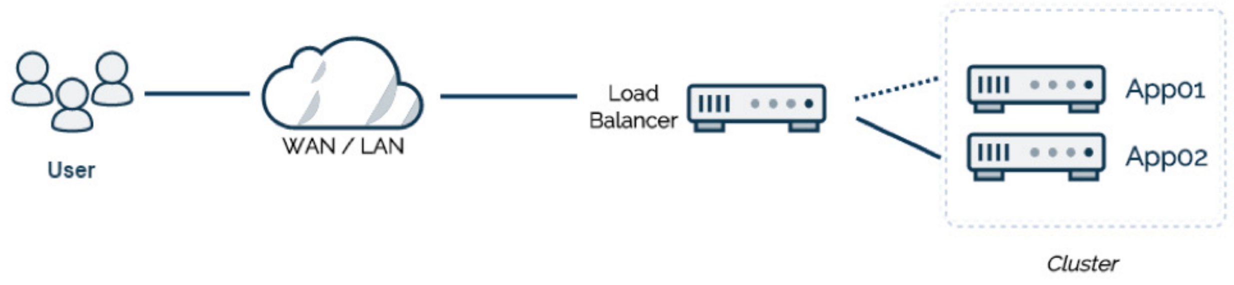
Vertical scaling is accomplished by increasing system resources, like adding more memory and processing power. Horizontal scaling, on the other hand, is accomplished by adding more servers to an existing cluster.

Let's talk about exactly what that means. Given the nature of ownCloud and PHP's out-of-the-box single threaded application design, ownCloud (as with most web-based PHP applications) performs / scales best in a clustered, or scaled-out environment. For this discussion, a cluster is simply a group of ownCloud servers. A load balancer distributes the workload between the ownCloud Enterprise servers in a cluster. At any point, an ownCloud Enterprise App server can be added to the existing cluster to handle more requests from users accessing your ownCloud Enterprise instance; this is horizontal scaling. This can be accomplished through the stateless operation of the ownCloud code.

While horizontal scaling is usually the most reliable and efficient method of scalability, it's not as trivial as vertical scaling. ownCloud Enterprise stores most of the configuration data in the database, so scaling out horizontally is extremely easy. Administrators normally install and successfully configure a single ownCloud virtual-server instance. Once that virtual-server is confirmed and tested, it is cloned using tools provided the underlying hypervisor, powered back up on the existing host or a separate host depending customer environment (a separate host provides minimal redundancy). Once the new virtual-server is running, a new IP address is given and the load balancer is configured with it.

The load balancer has a single responsibility: deciding which ownCloud Enterprise server from the cluster will receive a request from an end-user. It behaves like a reverse proxy, making the process seamless to the end-user. Least connections and round-robin are the two most common type of load balancer algorithms used in an ownCloud Enterprise solution. A load balancer with least connections combined with sticky sessions affords one of the simplest and most common types of load balancing in an ownCloud Enterprise solution. With least connections / sticky sessions the load balancer directs an initial end-user to the server with the least current connections and continues to send the user to the same server simplifying session management.

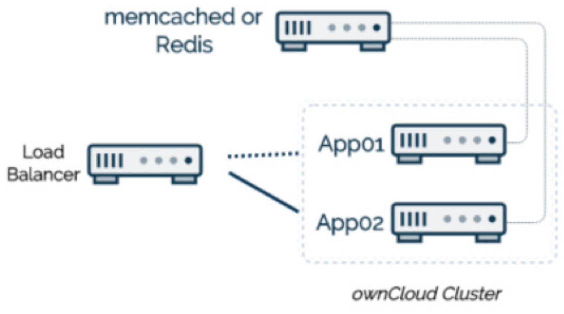
There are standard and enterprise-grade appliances from companies such as F5, A10, Kemp, etc. that are very fast and reliable and can also be set up in an N+1 environment, ensuring even higher availability. If a business is utilizing such enterprise-grade load-balancers they can and should be utilized in front of an ownCloud cluster, reducing deployment costs, setup, maintenance and complications. Open Source HAProxy is also a very popular and affordable option. Open Source HAProxy sits on top of a standard Linux based OS virtual-machine and can be easily configured, deployed and installed in an ownCloud Enterprise cluster. For High Availability solutions, HAProxy can be configured in an N+1 environment utilizing a heartbeat mechanism between two physically diverse HAProxys, ensuring uptime.



Horizontal Scaling with ownCloud

User Session Management and Persistence in an ownCloud Enterprise Cluster

ownCloud Enterprise traffic is based on user sessions and leverages PHP to store user session variables. Therefore when implementing an ownCloud Enterprise cluster behind a load balancer, session management and persistence must be taken into consideration. There are two fundamentally different ways to do session management in supporting an ownCloud cluster. One is local session management on the application servers combined with utilizing sticky / persistent sessions on the load balancer. The other is a centralized session management tool on the ownCloud Enterprise application virtual-servers. Business drivers will determine which session management solution is best, based on needs and after careful consideration of the pros and cons of each. For most business needs, local session management on the ownCloud Enterprise virtual-machine with the load balancer utilizing sticky / persistent sessions will suffice. If business needs dictate otherwise, Memcached or Redis solutions are fully supported by ownCloud and accommodate session management across multiple App servers in the cluster.



Distributed Session Management

ownCloud Enterprise, by design, does not store user session in the database. If ownCloud Enterprise has used the database, then session management across a cluster would not be an issue as the database would be a common storage point. Careful consideration was put into the design of user session storage within ownCloud. In order to enable ownCloud to scale beyond one half million users, ownCloud chose to implement in RAM session storage for speed and performance.

Transactional File Locking

ownClouds Transactional File Locking mechanism locks files to avoid file corruption during normal operations. Since this mechanism operates at a higher level than the filesystem it's not mandatory to use a filesystem that supports locking. This functionality is for example used to lock parent directories so that they can't be renamed during any activity on files inside these directories.

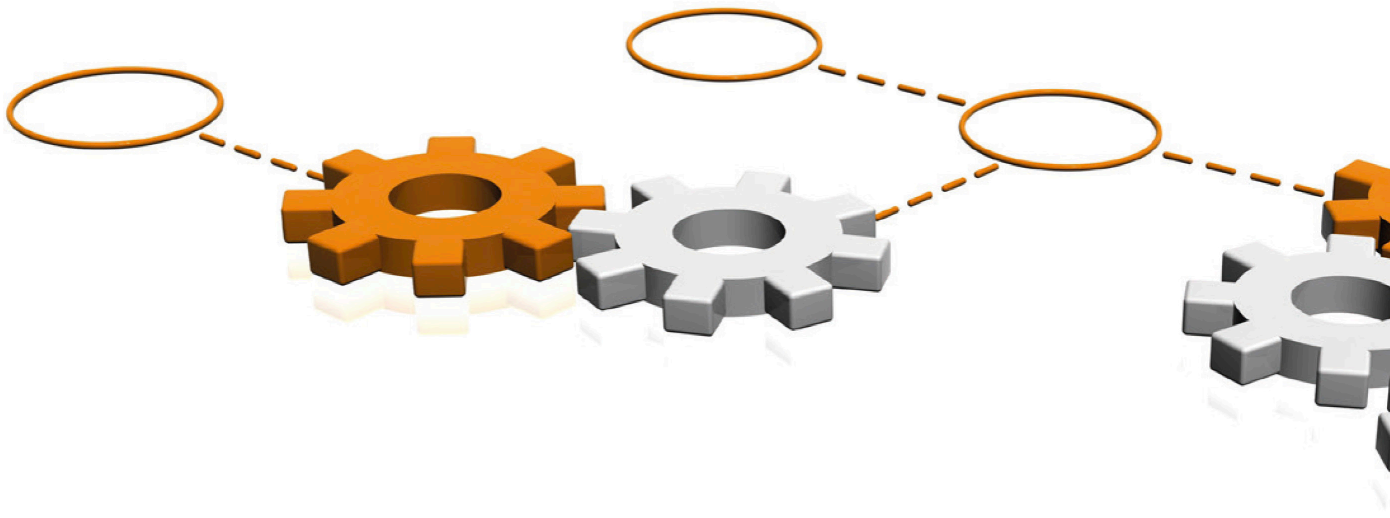
File Locking is enabled by default, using the database locking backend. Admins of ownCloud Servers with a heavy workload should install a Memcache to reduce the load on the database. To use a memcache with Transactional File Locking a Redis Server with a corresponding PHP module is needed.

All details can be found at docs.owncloud.com

PHP and memory caching

Some operations that an ownCloud Enterprise server executes take more time to complete, such as expensive calculations or communication with a remote storage server. Other operations are much faster – but are needed many times per second. To improve performance and reduce the load on the system caused by CPU / RAM intensive or frequently needed work, ownCloud Enterprise can cache the result of these operations. Caching is used in PHP to store compiled versions of the scripts so they don't need to be recompiled on every request. This is called Opcaching and has been included and enabled by default in PHP since the 5.5 release.

Memory caching, on the other hand, is used directly by web applications like ownCloud. It helps ownCloud avoid slow database queries or file system checks by retrieving a result from a memory cache, either on the local machine with APCu or distributed on a cluster of servers using Redis. The result is a trade-off of memory usage for improved performance. As the memory usage of these caches is typically small, it is generally worth the effort to set them up.



Server Architecture

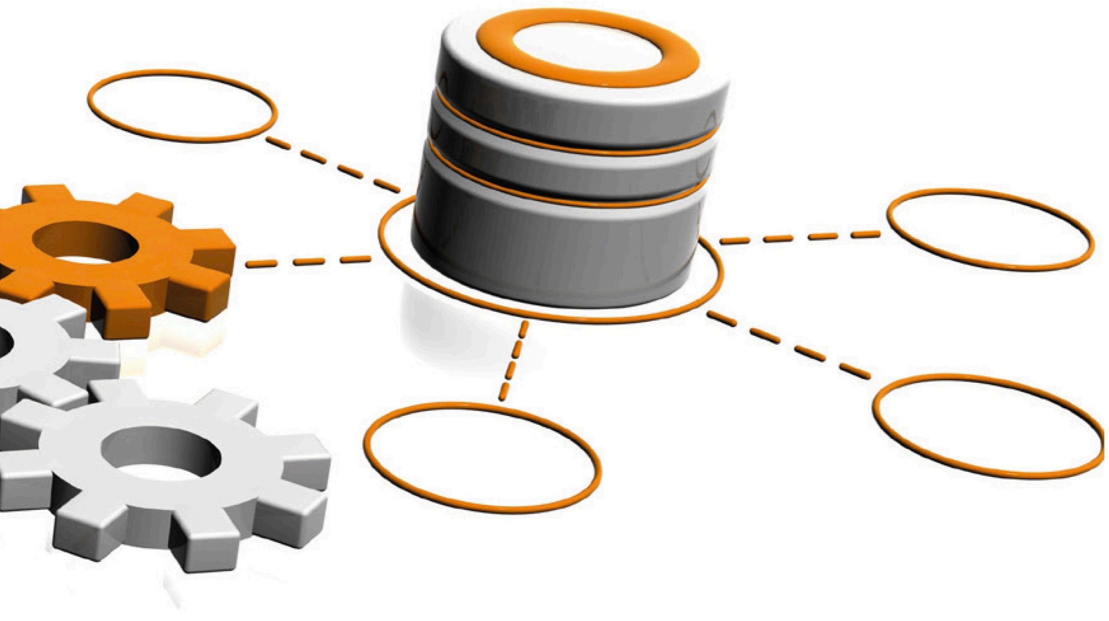
At its core, ownCloud Enterprise is a PHP web application running on top of Apache web servers, running on the customer‘s choice of supported Linux platforms. There is no advantage or disadvantage to running ownCloud on any of the supported Linux operating systems even if some distros may not provide the required and recommended PHP modules. For that reason, Ubuntu is one of the recommended distributions. In the case of selecting an OS for use with ownCloud, businesses will utilize what company policy dictates or leverage existing in-house knowledge to select the base OS. The ownCloud Enterprise PHP application manages every aspect of ownCloud Enterprise, from user management to plugins, file sharing and storage. Attached to the PHP application is a database where ownCloud Enterprise stores user information, user-shared file details, plug-in application states, and the ownCloud file cache.

ownCloud accesses the database through an abstraction layer, enabling support for Oracle, MySQL, and PostgreSQL. Complete webserver logging is provided via Web-server logs, and user and system logs are provided in a separate ownCloud Enterprise log, or can be directed to the Syslog file. Deploying, scaling, and maintaining ownCloud Enterprise is similar in fashion to any other LAMP-based web application and follows basic rules, practices and design principles commonly found in the industry.

Server Sizing

As with any HTTP-based LAMP deployment, ownCloud’s deployment scenarios are similar in configuration and resources. With ownCloud, the number of users (both total and concurrent), number of files attached / accessible to the ownCloud instance, available bandwidth, storage IOPs and CPU speed are all considerations when sizing your environment for production status.

Due to the many variables that are possible for each business and their solution, there is no right “one size fits all” deployment specification. Administrators will need to monitor, test, and possibly stress-test their particular deployment before a full production roll-out based on their utilization needs and hardware involved. We will explore a few example of deployments to give a better understanding of how ownCloud is deployed and right-sized, giving administrators a **base guideline** to designing the ownCloud Enterprise solution that best fits their business needs.



Implementation Examples

Company A

Company A is a 200-person company at one location with about 170 users identified as needing access to ownCloud Enterprise. Business need dictates that only Primary Storage is required and estimates about 11,000 files totaling 90GB need to be stored on the Primary Storage of ownCloud Enterprise. Company A has only one IT Administrator, on staff who has slight experience with Ubuntu and MySQL, thus Ubuntu is selected as the underlying operating system and Jane will utilize MySQL as the database provider.

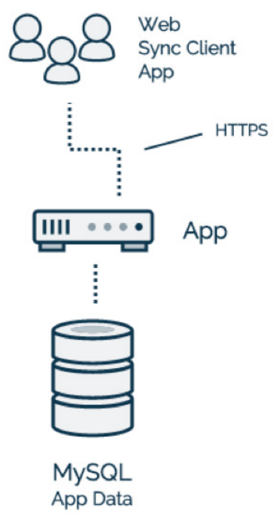
Company A currently is using virtualized technology based on VMware' ESXi hypervisor which resides on relatively current physical hosts of reasonable speed (RAM / CPU / Disk).

To deploy ownCloud Enterprise, Company A will create two virtual machines in their environment. One "APP" virtual-server with 2 vCPUs (2 vCPUs per host core) 8GB RAM and 150GB of disk space (30GB for the operating system and 120GB for user data, allowing for growth) and one "SQL" virtual-server, with 2 vCPUs, 8GB RAM and 20GB of disk space. On both virtual servers, they will install the latest stable version of a headless Ubuntu operating system.

Installing headless servers (i.e. minimal OS install without a GUI desktop) should be practiced with any web based application server. This not only minimizes hardware provision requirements (CPU, RAM and disk), but also reduces complexities and insecurities as well as ensuring uptime.

On the App virtual machine, Apache, and PHP along with any required dependencies will be installed utilizing normal practices for installing applications on a Linux server. Then the ownCloud Enterprise application is installed. Likewise, MySQL will be installed on the SQL virtual-server and configured following ownCloud Enterprise's instructions. Jane configures PHP to consume a maximum of 512MB and selects APCu as her PHP caching device and allocates 256MB to it.

Seeing that Company A requires public-facing access to their ownCloud environment, the App virtual-server will be provided with a public IP address (either by use of NAT at the firewall / router or assigned directly to the virtual-server) and set DNS entries respectively for the FQDN (Fully Qualified Domain Name) assigned. Company A's firewall will be configured to allow the App server to be in the DMZ and will open TCP port 443 and optional also port 80 access to the App server. As with any ownCloud installation, the SQL virtual-server does not need public IP access, so Jane assigns a private IP address (i.e. 10.10.64.128) and ensures that the App virtual-server can correctly communicate with the SQL virtual-server over TCP / UDP port MySQL (3306).



Architecture Example for Company A

Role	vCPU	RAM	PHP	PHP-Cache	OS / App	/data
App	2	8GB	512MB	256MB	30GB	120GB
SQL	2	8GB			20GB	

Company A installs the company's SSL certificates within Apache and confirms they are correctly installed, accessing the server with her web-browser. They also configure Apache to redirect all requests to the webserver on HTTP (port 80) to HTTPS (port 443) for the best experience for the end-users. After that, they can point a web-browser to the URL of their insntance in order to complete the initial configuration.

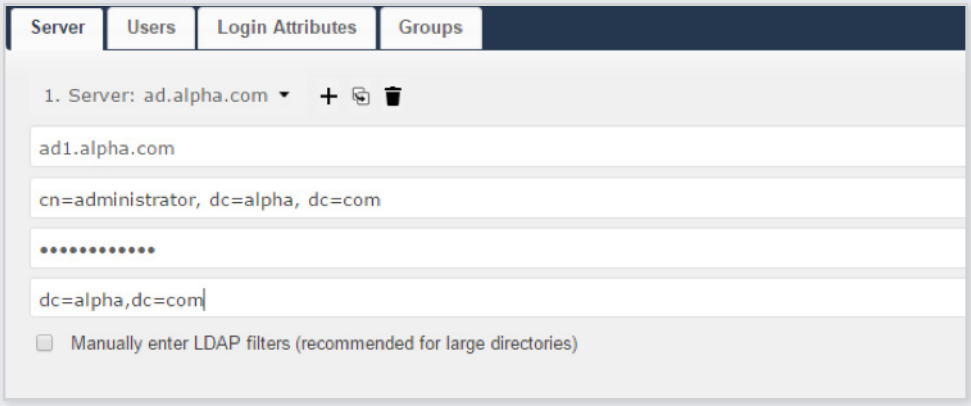
All client interfacing (Browser, Desktop Sync Client, Mobile Apps and WebDAV) to the ownCloud application is accomplished over HTTPS. No other ports need to be exposed to the internet. While access to the ownCloud server can be accomplished over HTTP (port 80), given today's CPU speed combined with security concerns, this is not recommended. Administrators should forward HTTP (port 80) to HTTPS (port 443) either at the firewall / router level or utilize Apache to forward port 80 for the best user experience.

Company A uses Microsoft Active Directory (AD) for user authentication. Once configured, administrators will select individual AD users or by AD groups, allowing access to the ownCloud application. Once correctly configured, Jane confirms the correct setup by successfully logging into the ownCloud web application utilizing her existing AD credentials.

Company A has a third-party onsite backup solution. Jane will install the subsequent Linux agent provided by the third-party vendor onto both the App and SQL virtual-machines and configure them to back up both the user /data directory and MySQL database. She will also make snapshots (as a form of backup) of the virtual-machines, as per IT policy, utilizing vSphere.

Once Jane ensures the ownCloud installation is operating correctly and secure, she deploys to a small subset of "power-users" within Company A, for initial testing and feedback. Once initial testing is complete and feedback is satisfactory, she deploys the ownCloud Desktop Sync Client, ownCloud Mobile application as required. After training users on the use of ownCloud, she will monitor the server and environment as she would with any other web application.

ownCloud utilizes two forms of user authentication: external (LDAP, Active Directory,SAML 2.0 or OpenID Connect) and internal (stored in database). Both external and internal authentication can exist simultaneously. For example, a business can integrate ownCloud with Active Directory to allow authentication for their current employees' access, and at the same time can utilize ownCloud Enterprise's internal user authentication mechanism for outside contractors' access. Contractors that do not have an AD account but require access to files located on the ownCloud server shares. ownCloud Clients like the mobile Apps (iOS/Android) and the Desktop Client can be authenticated using basic auth with username/ password or OAuth2.



1. Server: ad.alpha.com

ad1.alpha.com

cn=administrator, dc=alpha, dc=com

••••••••••••

dc=alpha,dc=com

☐ Manually enter LDAP filters (recommended for large directories)

LDAP Configuration in ownCloud

Company B

Company B is a 450-person Engineering company with approximately 400 users currently approved to have access to the ownCloud Enterprise environment. Company B is growing rapidly so they want to implement without creating any bottlenecks with their growth. Company B has three different locations, utilizing a hub and spoke IT design with good bandwidth between each.

Company B has existing Windows Server shares to integrate into the ownCloud Enterprise environment and is not looking to utilize ownCloud Enterprise's Primary Storage extensively. Throughout the Windows Server shares across multiple servers, there are 110,000 files consuming 1.2TB of storage. Company B has an administrator who is familiar with CentOS and MySQL. To deploy ownCloud Enterprise, Company B administrators will initially create three virtual-machines in their environment.

Two "APP" virtual-servers with 4 vCPUs (4 vCPUs per host core) 16GB RAM and 30GB of disk space, one "SQL" virtual-server, with 4 vCPUs, 16GB RAM and 40GB of disk space and one "LB" (load Balancer) with 1 vCPU, 8GB RAM and 15GB disk space. They will install the latest stable version of a headless CentOS operating system on all of their virtual servers. The SAN administrator will create a 10GB NFS partition on the company's SAN, allowing the Linux administrator to mount the NFS to each ownCloud Enterprise App server utilizing the FStab mounting scheme. This NFS mount will accommodate ownCloud Enterprise's /data directory. Even though Company B will use Windows Server shares utilizing ownCloud Enterprise's WND app for their data storage, the /data directory is still required for proper operation. In this case, 25MB per user (plus overhead) was used to calculate the size of the NFS mount.

While ownCloud will utilize this user /data directory for metadata such as thumbnails, versioning, and trash-bin, it is recommended that the ownCloud quota feature be implemented and set at 25MB per user. With a quota specified, ownCloud will ensure that the user directory will comply with the induced quota by maintaining the individual users trash-bin and versioning directories (oldest out).

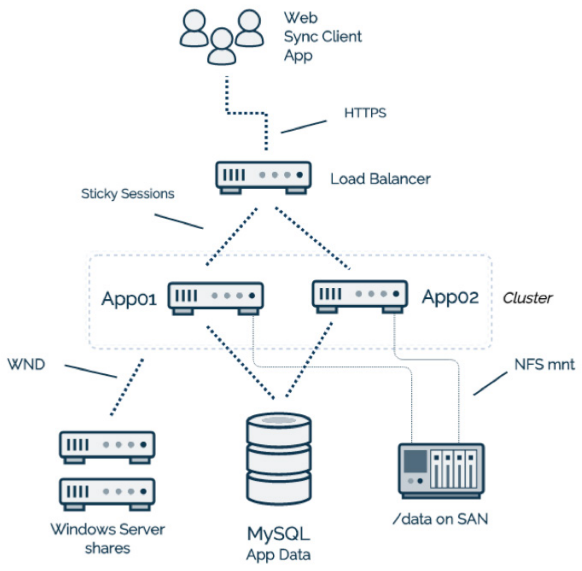
Role	vCPU	RAM	PHP	PHP-Cache	OS / App	/data
LB	1	8GB			15GB	
App (2)	4	16GB	512MB	512MB	30GB	10GB
SQL	4	16GB			40GB	

Sizing Examples for Company B

Company B has no current load balancer in place so HAProxy was chosen for speed, performance and budgeting concerns and will be installed on the appropriate virtual machine. Once installed, HAProxy will be configured to point to the IP addresses assigned to the two ownCloud Enterprise App virtual-servers utilizing the Least Connection format (with Round Robin being an option) and Sticky Sessions enabled.

Installation of the application server will consist of installing Apache, PHP, APCu for PHP caching and ownCloud Enterprise. Once this virtual-server is fully configured (ownCloud setting, PHP settings, Apache configuration and SSL certificates etc.), tested and proven, it will be cloned using tools provided within the underlying hypervisor, IP address re-configured accordingly and powered up into the cluster. MySQL is installed and configured accordingly on the SQL virtual-machine. Redis should be configured on a dedicated server and used for distributed caching and Transactional File Locking.

While the two App virtual-servers can be installed on separate physical hosts this does not provide a true HA (High Availability) solution given that the SQL and Load Balancer are singular. Tools such as VMware HA and DRS can be called upon to reduce outage times. If a true, more hardened, HA solution is required, please see the next sample solution.



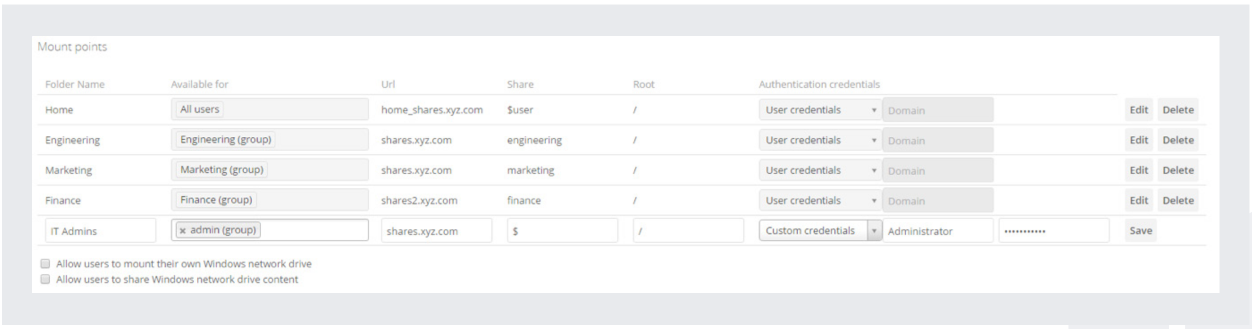
Architecture Example for Company B

HA, or High Availability solutions refer to solutions with automated fail-over and recovery, while FT or Fault Tolerant ones ensure zero downtime. Company B has opted to utilize existing Windows server shares due to the extensive ACLs in place along with the desire not to have data reside in two places (data overlap). ownCloud Enterprise administrators will configure the Windows Network Drive app of ownCloud Enterprise to point to each specific existing Windows share. Company B has decided to share not only the users‘ existing home directory, but specific department shares as well.

Since Company B has also integrated ownCloud Enterprise into their existing Microsoft AD, Company B can also leverage ownCloud Enterprise’s ability to use the users‘ login credentials. Using the login credentials allows ownCloud to honor existing Windows ACLs. ownCloud Enterprise end-users will see only what their AD login credentials allow.

Once the initial install is completed, tested, verified and the company mandated testing is completed, the IT manager will deploy to a small test group compromising of IT staff and other power-users. During this test period, systems will be monitored and gauged for effectiveness. Once the IT staff is satisfied with the results, they will deploy the desktop sync clients, mobile apps and train the employees in groups of 100 with three day lapse in between the next group of 100. During each segmented deployment, IT staff is monitoring the ownCloud Enterprise systems and making any minor changes if required.

Initial deployment in an ownCloud Enterprise solution is normally when you see the most load or stress, due to the number of new users and new sync requests. In this case, the IT Manager decided to segment the deployment to relieve any stresses that might have slowed the systems and provided new ownCloud Enterprise users with the best possible experience with the new solution.
The other option that the IT Manager had at his disposal was to increase the number of App servers to handle the initial deployment of all 400 users at once. Once the initial deployment is completed, monitoring the systems would ensure administrators the timely removal of the additional servers deployed during initial deployment.



Mount points

Folder Name	Available for	Url	Share	Root	Authentication credentials				
Home	All users	home_shares.xyz.com	$user	/	User credentials	Domain		Edit	Delete
Engineering	Engineering (group)	shares.xyz.com	engineering	/	User credentials	Domain		Edit	Delete
Marketing	Marketing (group)	shares.xyz.com	marketing	/	User credentials	Domain		Edit	Delete
Finance	Finance (group)	shares2.xyz.com	finance	/	User credentials	Domain		Edit	Delete
IT Admins	× admin (group)	shares.xyz.com	$	/	Custom credentials	Administrator	•••••••••••	Save	

☐ Allow users to mount their own Windows network drive
☐ Allow users to share Windows network drive content

Accessing Windows Network Drives with ownCloud

Company C

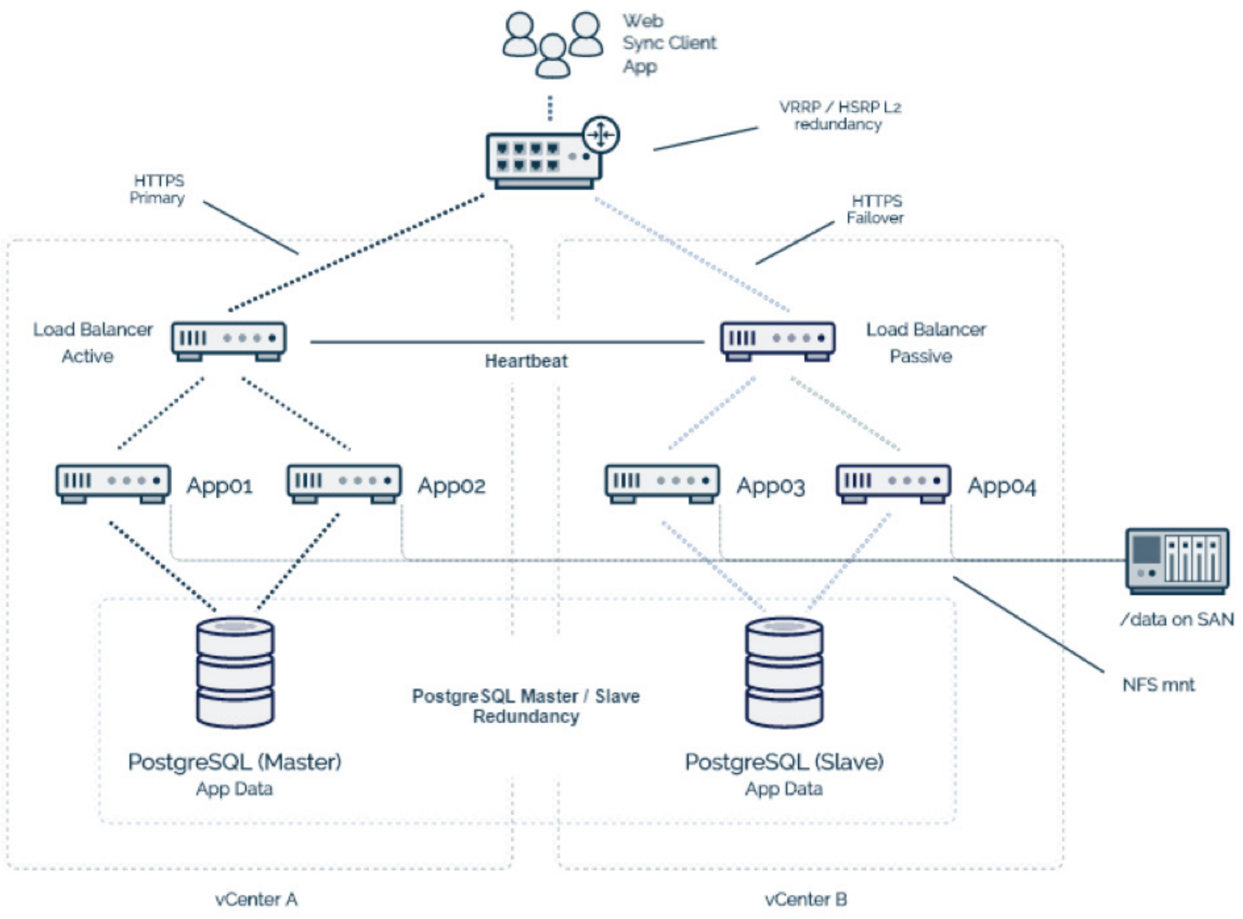
Company C is a growing financial institution with approximately 1100 employees, of which 800 employees are required to have access to the ownCloud Enterprise solution. Due to the nature and time constraints of the financial transactions, they need a Fault Tolerant (FT) solution. Business drivers dictate the need for a 99.999% uptime ("5 nines") solution for the ownCloud Enterprise environment. They have a single data center / hub with multiple locations / spokes throughout all four time zones in the continental US. Within this data center, they utilize two hypervisor vCenters for redundancy and Company C plans to leverage these for ownCloud Enterprise redundancy.

Due to the number of possible concurrent users utilizing the ownCloud service, Company C plans to utilize two ownCloud Enterprise web application virtual-servers for each vCenter (four ownCloud Enterprise App servers total due to redundancy). They have no current Windows shares that would be useful for their end-users so they elect to use ownCloud Enterprise's Primary Storage as the default storage.

Role	vCPU	RAM	PHP	PHP-Cache	OS / App	/data
LB (2)	1	8GB			15GB	
App (4)	4	16GB	1GB	1GB	30GB	4.7TB
SQL (2)	4	16GB			40GB	

Sizing Examples for Company C

Company C already has a substantial investment in an existing highly redundant SAN, so they will create an NFS mount point for the ownCloud /data directory. After speaking to management and considering all business drivers, they have elected to allocate 5GB of disk space for each ownCloud user and the SAN-admin creates a 4.7TB NFS mount point on their company's SAN.



Architecture Example for Company C

Company C has standardized on Red Hat as their primary Linux operating system and due to in-house expertise, selected PostgreSQL as the database software.

Initially, they will create 3 virtual machines; one each for the Load Balancer, APP and MySQL. The Linux Administrator will install HAProxy on the Load Balancer virtual-machine, ownCloud Enterprise on the App server and MySQL on the SQL virtual-machine. The Linux Administrator will also mount the NFS created by the SAN administrator and modify ownCloud to point to the new /data directory. Company C has also elected to install Redis on the App virtual-machine for file-locking and Redis for both file locking and Redis with 1GB of RAM for improved PHP caching performance. Once Company C has all three virtual-machines (load balancer, App and SQL) in place, it will begin its testing.

Once satisfied with the performance, it will „clone“ the single Aplication-Server instance and bring up a second Aplication-Server, configure both and update the Load Balancer accordingly. Again testing takes place. Once fully tested, IT administrators will replicate the solution (Load Balancer, two Application-Servers , and SQL virtual-machines) onto a separate vCenter for redundancy. A heartbeat will be setup between the two load balancers, with one set to active and the other to passive. A Master / Slave Relationship will be established between the two PostgreSQL virtual-machines.

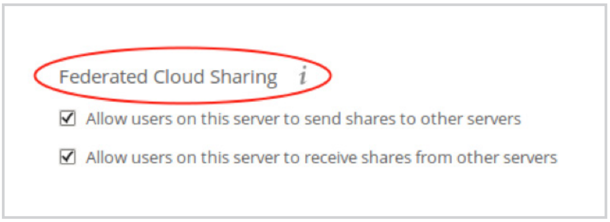
Master / Master and Master / Slave relationships of the SQL servers both allow to be configured for a transparent failover. In case of an Master / Master setup it´s mandatory to configure a SQL Proxy like ProxySQL, Maxscale or F5-Splitter to avoid deathlocks.

Company D

Big Corporate Conglomerate (BCC) is a large 15,000 employee company with many diverse business units (i.e. Retail, government contracting, banking, etc.). From a business standpoint, these divisions operate independently with minimal need to communicate with other divisions, and some modest requirement to communicate with their corporate headquarters.

After a careful review off the business needs within BCC and developing a solution for an Enterprise File Sync and Share, they have settled on utilizing ownCloud Enterprise for their needs. Within BCC, the need to collaborate between divisions is minimal. Of particular interest was the capability of BCC not only to create individual / separate ownCloud Enterprise solutions for each division, but also to utilize ownCloud Enterprise's Federated Cloud sharing to accommodate the lesser need of collaborating amongst divisions. Headquarters also has a vital need to securely receive financial reports, marketing documents as well as other files securely and seamlessly from each division.

With Federated Cloud, users at one division of BCC can collaborate with users at another division while each divisional ownCloud Enterprise server maintains its respective security and governance protocols and does not require BCC to replicate unneeded data across diverse business entities.


Federated Cloud Sharing

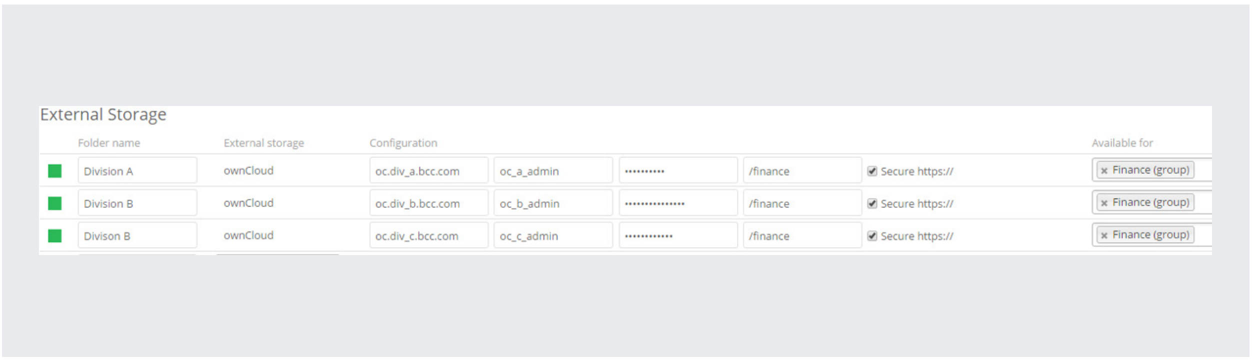
☑ Allow users on this server to send shares to other servers

☑ Allow users on this server to receive shares from other servers

Federated Cloud Sharing Options

ownCloud Federation gives users the flexibility and transparency to securely and easily share files between divisions without IT administrator involvement. BCC users are no longer confined to a single shared folder or servers within their respective divisions.

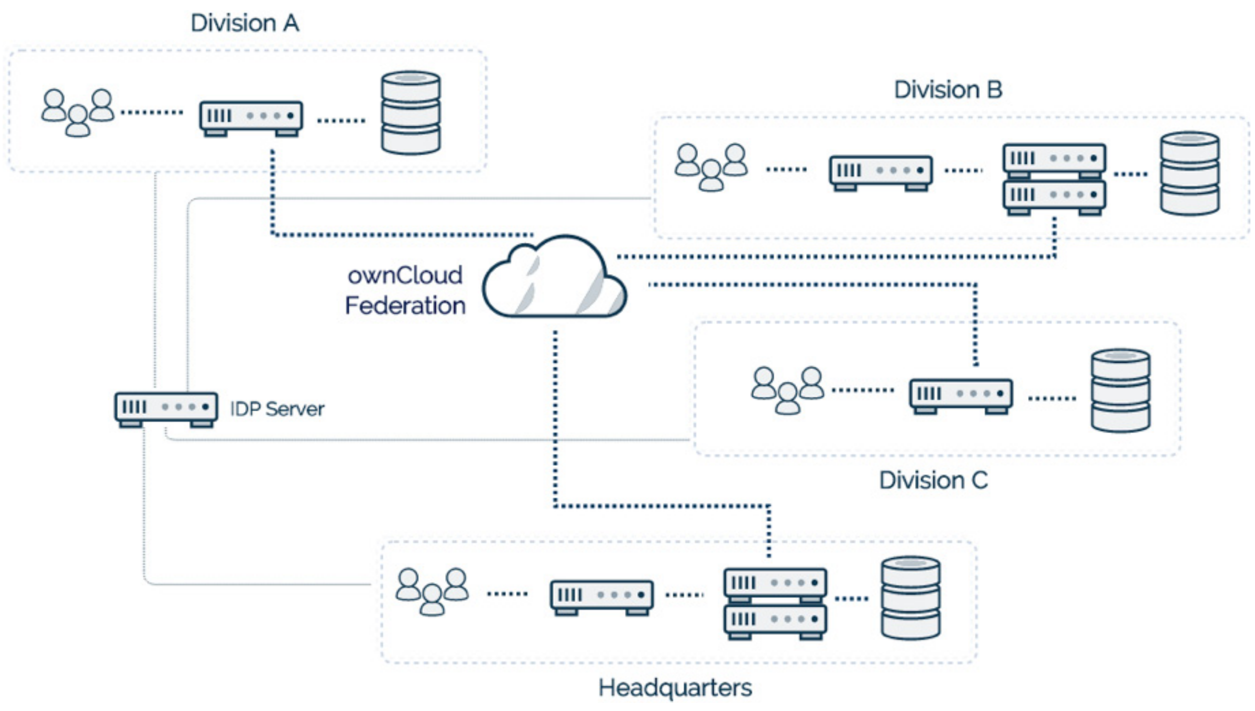
Federated sharing with ownCloud Enterprise also gives BCC the ability for administrators to mount commonly used data through the use of the External Storage App. This, combined with the ability for end-users to individually share files and folders with other division users, will provide BCC the collaboration it needs across different division silos. BCC has a corporate security policy that requires the use of two-factor authentication for all users. BCC selected ownCloud Enterprise to accomplish these goals. BCC will install the ownCloud Enterprise Shibboleth app and install and configure mod_shib along with Apache on their environment. BCC will configure both the `/etc/apache2/conf.d/shib.conf`


External Storage

	Folder name	External storage	Configuration					Available for
🟩	Division A	ownCloud	oc.div_a.bcc.com	oc_a_admin	••••••••••	/finance	☑ Secure https://	× Finance (group)
🟩	Division B	ownCloud	oc.div_b.bcc.com	oc_b_admin	•••••••••••••••	/finance	☑ Secure https://	× Finance (group)
🟩	Divison B	ownCloud	oc.div_c.bcc.com	oc_c_admin	••••••••••••	/finance	☑ Secure https://	× Finance (group)

Mounting Federated Cloud Shared as External Storage to ownCloud and the `/var/www/html/owncloud/config/config.php` on each ownCloud App virtual machine. Once configured, BCC will test each server to ensure proper configuration and performance.

To deploy the Federated ownCloud Enterprise solution company-wide, BCC will treat each diverse division unit as a separate solution as far as the ownCloud Enterprise solution is concerned. Given the different technologies in place for each division, BCC IT Managers will customize each solution to connect to the resources within each division, but attempt to keep common practices within each environment (i.e Linux OS, WWW Server, Load balancer, SQL server, etc.) as required. BCC can use the samples provided here as a baseline for each divisional deployment.



Architecture Example for Company D

About ownCloud

ownCloud is the market leading open source content collaboration solution worldwide. ownCloud enables users to securely access and share data from any device, anywhere in the world. With more than 200,000 installations and 50 million users, ownCloud provides organizations a modern collaborative experience, thereby boosting productivity without compromising on security. At the same time, it gives organizations the visibility and control required to manage sensitive data.

To get the latest updates, please visit https://owncloud.com/newsroom or follow us on Twitter @ownCloud.

ownCloud Architectural Guide EN 191107

ownCloud GmbH
Rathsbergstr. 17
90411 Nürnberg
Germany

Contact:
owncloud.com/contact
Phone: +49 911 14888690
owncloud.com

@ownCloud
facebook.com/owncloud
linkedin.com/company/owncloud