Instalação e Configurações Nextcloud Docker

Procedimentos de instalação Nextcloud Docker

- Nextcloud Docker oficial instalação
- Nextcloud LDAP authentication
- Remove Nextcloud Share from Database

Nextcloud Docker oficial instalação

Link: https://github.com/nextcloud/docker

18/05/2025

What is Nextcloud?

A safe home for all your data. Access & share your files, calendars, contacts, mail & more from any device, on your terms.

logo

⚠⚠⚠ This image is maintained by community volunteers and designed for expert use. For quick and easy deployment that supports the full set of Nextcloud Hub features, use the Nextcloud All-in-One docker container maintained by Nextcloud GmbH.

How to use this image

This image is designed to be used in a micro-service environment. There are two versions of the image you can choose from.

The `apache` tag contains a full Nextcloud installation including an apache web server. It is designed to be easy to use and gets you running pretty fast. This is also the default for the `latest` tag and version tags that are not further specified.

The second option is a `fpm` container. It is based on the php-fpm image and runs a fastCGI-Process that serves your Nextcloud page. To use this image it must be combined with any webserver that can proxy the http requests to the FastCGI-port of the container.

Try in PWD

Getting help

Most Nextcloud Server administrative matters are covered in the official Nextcloud Admin Manual or other official Nextcloud documentation (which are all routinely updated).

Discourse Users Discourse Posts

If you have any problems or usage questions while using the image, please ask for assistance on the Nextcloud Community Help Forum rather than reporting them as "bugs" (unless they are bugs of course). This helps the maintainers (who are volunteers) remain focused on making the image better (rather than responding solely to one-on-one support issues). (Tip: Some of the maintainers are also regular responders to help requests on the community help forum.)

For the image specifically, we provide some simple deployment examples as well as some more extensive deployment examples. In addition, the community help forum has a "how-to" section with further examples of other peoples' container based Nextcloud stacks.

Below you'll find the main documentation for using this image.

Using the apache image

The apache image contains a webserver and exposes port 80. To start the container type:

```
$ docker run -d -p 8080:80 nextcloud
```

Now you can access Nextcloud at http://localhost:8080/ from your host system.

WARNING: This example is only suitable for limited testing purposes. Please read on to understand how the image handles storing your data and other aspects you need to consider to establish a full Nextcloud stack.

Using the fpm image

To use the fpm image, you need an additional web server, such as nginx, that can proxy http-request to the fpm-port of the container. For fpm connection this container exposes port 9000. In most cases, you might want to use another container or your host as proxy. If you use your host you can address your Nextcloud container directly on port 9000. If you use another container, make sure that you add them to the same docker network (via `docker run --network <NAME> ...` or a `docker compose` file). In both cases you don't want to map the fpm port to your host.

```
$ docker run -d nextcloud:fpm
```

As the fastCGI-Process is not capable of serving static files (style sheets, images, ...), the webserver needs access to these files. This can be achieved with the `volumes-from` option. You can find more information in the docker compose section.

Using an external database

By default, this container uses SQLite for data storage but the Nextcloud setup wizard (appears on first run) allows connecting to an existing MySQL/MariaDB or PostgreSQL database. You can also

link a database container, e. g. `--link my-mysql:mysql`, and then use `mysql` as the database host on setup. More info is in the docker compose section.

Persistent data

The Nextcloud installation and all data beyond what lives in the database (file uploads, etc.) are stored in the unnamed docker volume volume `/var/www/html`. The docker daemon will store that data within the docker directory `/var/lib/docker/volumes/...`. That means your data is saved even if the container crashes, is stopped or deleted.

A named Docker volume or a mounted host directory should be used for upgrades and backups. To achieve this, you need one volume for your database container and one for Nextcloud.

Nextcloud:

- `/var/www/html/` folder where all Nextcloud data lives

```
$ docker run -d \
-v nextcloud:/var/www/html \
nextcloud
```

Database:

- `/var/lib/mysql` MySQL / MariaDB Data
- `/var/lib/postgresql/data` PostgreSQL Data

```
$ docker run -d \
-v db:/var/lib/mysql \
mariadb:lts
```

Additional volumes

If you want to get fine grained access to your individual files, you can mount additional volumes for data, config, your theme and custom apps. The `data`, `config` files are stored in respective subfolders inside `/var/www/html/`. The apps are split into core `apps` (which are shipped with Nextcloud and you don't need to take care of) and a `custom_apps` folder. If you use a custom theme it would go into the `themes` subfolder.

Overview of the folders that can be mounted as volumes:

- `/var/www/html` Main folder, needed for updating
- `/var/www/html/custom_apps` installed / modified apps
- `/var/www/html/config` local configuration
- `/var/www/html/data` the actual data of your Nextcloud
- `/var/www/html/themes/<YOUR_CUSTOM_THEME>` theming/branding

If you want to use named volumes for all of these, it would look like this:

```
$ docker run -d \
-v nextcloud:/var/www/html \
-v custom_apps:/var/www/html/custom_apps \
-v config:/var/www/html/config \
-v data:/var/www/html/data \
-v theme:/var/www/html/themes/<YOUR_CUSTOM_THEME> \
nextcloud
```

If you'd prefer to use bind mounts instead of named volumes, for instance, when working with different device or network mounts for user data files and configuration:

```
$ docker run -d \
-v /path/on/host/to/folder/nextcloud:/var/www/html \
-v /path/on/host/to/folder/custom_apps:/var/www/html/custom_apps \
-v /path/on/host/to/folder/config:/var/www/html/config \
-v /path/on/host/to/folder/data:/var/www/html/data \
-v /path/on/host/to/folder/theme:/var/www/html/themes/<YOUR_CUSTOM_THEME> \
nextcloud
```

Here’s the same example using Docker's more detailed `--mount`. Note that with `-v` or `--volume`, the specified folders are created automatically if they don't exist. However, when using `--mount` for bind mounts, the directories must already exist on the host, or Docker will return an error.

```
$ docker run -d \
--mount type=bind,source=/path/on/host/to/folder/nextcloud,target=/var/www/html \
--mount type=bind,source=/path/on/host/to/folder/custom_apps,target=/var/www/html/custom_apps \
--mount type=bind,source=/path/on/host/to/folder/config,target=/var/www/html/config \
--mount type=bind,source=/path/on/host/to/folder/data,target=/var/www/html/data \
--mount type=bind,source=/path/on/host/to/folder/theme,target=/var/www/html/themes/<YOUR_CUSTOM_
nextcloud
```

The examples above use figurative directory `/path/on/host/to/folder/` for bind mounts. Please modify the paths by using either a relative or absolute path.

NOTE: Do not confuse the `apps` and `custom_apps` folders. These folders contain different sets of apps, and mixing them will result in a broken installation. The former contains "shipped" apps, which come with Nextcloud Server. The latter contains apps you install from the App Store.

Custom volumes

If mounting additional volumes under `/var/www/html`, you should consider:

- Confirming that upgrade.exclude contains the files and folders that should persist during installation and upgrades; or
- Mounting storage volumes to locations outside of `/var/www/html`.

Data inside the main folder (`/var/www/html`) will be overridden/removed during installation and upgrades, unless listed in upgrade.exclude. The additional volumes officially supported are already in that list, but custom volumes will need to be added by you. We

suggest mounting custom storage volumes outside of `/var/www/html` and if possible read-only so that making this adjustment is unnecessary. If you must do so, however, you may build a custom image with a modified `/upgrade.exclude` file that incorporates your custom volume(s).

Running as an arbitrary user / file permissions / changing the default container user

The default user within a container is root (uid = 0). By default, processes inside the container will expect to have root privileges. Network services will drop privileges and use `www-data` to serve requests.

Depending on your volumes configuration, this can lead to permission issues. You can address this by running the container with a different default user. When changing the default user, the image will no longer assume it has root privileges and will run all processes under the specified uid. To accomplish this, use the `--user` / `user` option in your container environment.

See:

- https://docs.docker.com/engine/containers/run/#user
- https://github.com/docker-library/docs/tree/master/php#running-as-an-arbitrary-user
- https://docs.podman.io/en/stable/markdown/podman-run.1.html#user-u-user-group

Accessing the Nextcloud command-line interface (`occ`)

To use the Nextcloud command-line interface (aka. `occ` command):

```
$ docker exec -it --user www-data CONTAINER_ID php occ
```

or for docker compose:

```
$ docker compose exec --user www-data app php occ
```

or even shorter:

```
$ docker compose exec -u33 app ./occ
```

Note: substitute `82` for `33` if using the Alpine-based images.

Viewing the Nextcloud configuration (`config.php`)

The image takes advantage of Nextcloud's Multiple config.php support to inject auto configuration environment variables and set image specific config values.

This means that merely viewing your `config.php` will not give you an accurate view of your running config. Instead, you should use Nextcloud's `occ config:list system` command to get get a

complete view of your merged configuration. This has the added benefit of automatically omitting sensitive values such as passwords and secrets from the output by default (e.g. useful for shared publicly or assisting others when troubleshooting or reporting a bug).

```
$ docker compose exec -u33 app ./occ config:list system
```

The `--private` flag can also be specified, in order to output all configuration values including passwords and secrets.

Auto configuration via environment variables

The Nextcloud image supports auto configuration of the Nextcloud Server installation via environment variables. You can preconfigure everything that would otherwise be prompted for by the Nextcloud Installation Wizard (as well as a few other key parameters relevant to initial installation).

Database parameters

To enable auto configuration, define your database connection via the following environment variables. If you set any group of values (i.e. all of `MYSQL_DATABASE`, `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_HOST`), they will not be requested via the Installation Wizard on first run.

You must specify all of the environment variables for a given database or the database environment variables defaults to SQLITE. ONLY use one database type!

SQLite:

- `SQLITE_DATABASE` Name of the database using sqlite

MYSQL/MariaDB:

- `MYSQL_DATABASE` Name of the database using mysql / mariadb.
- `MYSQL_USER` Username for the database using mysql / mariadb.
- `MYSQL_PASSWORD` Password for the database user using mysql / mariadb.
- `MYSQL_HOST` Hostname of the database server using mysql / mariadb.

PostgreSQL:

- `POSTGRES_DB` Name of the database using postgres.
- `POSTGRES_USER` Username for the database using postgres.
- `POSTGRES_PASSWORD` Password for the database user using postgres.
- `POSTGRES_HOST` Hostname of the database server using postgres.

As an alternative to passing sensitive information via environment variables, `_FILE` may be appended to the previously listed environment variables, causing the initialization script to load the values for those variables from files present in the container. See Docker secrets section below for details.

Initial admin account

If you specify all the variables for your database type (above), you can also auto configure your initial admin user and password (only works if you set both):

- `NEXTCLOUD_ADMIN_USER` Name of the Nextcloud admin user.
- `NEXTCLOUD_ADMIN_PASSWORD` Password for the Nextcloud admin user.

Specifying a complete database and admin credential config set at initial deployment will result in a fully automated installation (i.e. bypassing the web-based Nextcloud Installation Wizard).

Addition parameters may also be set at installation time and are described below.

Custom Data directory (`datadirectory`)

If you don't want to use the default data directory (`datadirectory`) location, you can set a custom one:

- `NEXTCLOUD_DATA_DIR` (default: `/var/www/html/data`) Configures the data directory where nextcloud stores all files from the users.

Trusted domains (`trusted_domains`)

One or more trusted domains can be set through environment variable, too. They will be added to the configuration after install.

- `NEXTCLOUD_TRUSTED_DOMAINS` (not set by default) Optional space-separated list of domains

Image specific

The install and update script is only triggered when a default command is used (`apache-foreground` or `php-fpm`). If you use a custom command you have to enable the install / update with

- `NEXTCLOUD_UPDATE` (default: `0`)

You might want to make sure the htaccess is up to date after each container update. Especially on multiple swarm nodes as any discrepancy will make your server unusable.

- `NEXTCLOUD_INIT_HTACCESS` (not set by default) Set it to true to enable run `occ maintenance:update:htaccess` after container initialization.

Redis Memory Caching

To use Redis for memory caching as well as PHP session storage, specify the following values and also add a Redis container to your stack. See the examples for further instructions.

- `REDIS_HOST` (not set by default) Name of Redis container

- `REDIS_HOST_PORT` (default: `6379`) Optional port for Redis, only use for external Redis servers that run on non-standard ports.
- `REDIS_HOST_USER` (not set by default) Optional username for Redis, only use for external Redis servers that require a user.
- `REDIS_HOST_PASSWORD` (not set by default) Redis password

Check the Nextcloud documentation for more information.

E-mail (SMTP) Configuration

To use an external SMTP server, you have to provide the connection details. Note that if you configure these values via Docker, you should **not** use the Nexcloud Web UI to configure external SMTP server parameters. Conversely, if you prefer to use the Web UI, do **not** set these variables here (because these variables will override whatever you attempt to set in the Web UI for these parameters). To configure Nextcloud to use SMTP add:

- `SMTP_HOST` (not set by default): The hostname of the SMTP server.
- `SMTP_SECURE` (empty by default): Set to `ssl` to use SSL, or `tls` to use STARTTLS.
- `SMTP_PORT` (default: `465` for SSL and `25` for non-secure connections): Optional port for the SMTP connection. Use `587` for an alternative port for STARTTLS.
- `SMTP_AUTHTYPE` (default: `LOGIN`): The method used for authentication. Use `PLAIN` if no authentication is required.
- `SMTP_NAME` (empty by default): The username for the authentication.
- `SMTP_PASSWORD` (empty by default): The password for the authentication.
- `MAIL_FROM_ADDRESS` (not set by default): Set the local-part for the 'from' field in the emails sent by Nextcloud.
- `MAIL_DOMAIN` (not set by default): Set a different domain for the emails than the domain where Nextcloud is installed.

At a minimum, the `SMTP_HOST`, `MAIL_FROM_ADDRESS` and `MAIL_DOMAIN` parameters must be defined.

Check the Nextcloud documentation for other values to configure SMTP.

Object Storage (Primary Storage)

By default, Nextcloud stores all files in `/var/www/html/data/` (or whatever custom data directory you've configured). Nextcloud also allows the use of object storages (like OpenStack Swift or any Amazon S3-compatible implementation) as *Primary Storage*. This semi-replaces the default storage of files in the data directory. Note: This data directory might still be used for compatibility reasons and still needs to exist. Check the Nextcloud documentation for more information.

To use an external S3 compatible object store as primary storage, set the following variables:

- `OBJECTSTORE_S3_BUCKET`: The name of the bucket that Nextcloud should store the data in
- `OBJECTSTORE_S3_REGION`: The region that the S3 bucket resides in
- `OBJECTSTORE_S3_HOST`: The hostname of the object storage server

- `OBJECTSTORE_S3_PORT`: The port that the object storage server is being served over
- `OBJECTSTORE_S3_KEY`: AWS style access key
- `OBJECTSTORE_S3_SECRET`: AWS style secret access key
- `OBJECTSTORE_S3_STORAGE_CLASS`: The storage class to use when adding objects to the bucket
- `OBJECTSTORE_S3_SSL` (default: `true`): Whether or not SSL/TLS should be used to communicate with object storage server
- `OBJECTSTORE_S3_USEPATH_STYLE` (default: `false`): Not required for AWS S3
- `OBJECTSTORE_S3_LEGACYAUTH` (default: `false`): Not required for AWS S3
- `OBJECTSTORE_S3_OBJECT_PREFIX` (default: `urn:oid:`): Prefix to prepend to the fileid
- `OBJECTSTORE_S3_AUTOCREATE` (default: `true`): Create the container if it does not exist
- `OBJECTSTORE_S3_SSE_C_KEY` (not set by default): Base64 encoded key with a maximum length of 32 bytes for server side encryption (SSE-C)

Check the Nextcloud documentation for more information.

To use an external OpenStack Swift object store as primary storage, set the following variables:

- `OBJECTSTORE_SWIFT_URL`: The Swift identity (Keystone) endpoint
- `OBJECTSTORE_SWIFT_AUTOCREATE` (default: `false`): Whether or not Nextcloud should automatically create the Swift container
- `OBJECTSTORE_SWIFT_USER_NAME`: Swift username
- `OBJECTSTORE_SWIFT_USER_PASSWORD`: Swift user password
- `OBJECTSTORE_SWIFT_USER_DOMAIN` (default: `Default`): Swift user domain
- `OBJECTSTORE_SWIFT_PROJECT_NAME`: OpenStack project name
- `OBJECTSTORE_SWIFT_PROJECT_DOMAIN` (default: `Default`): OpenStack project domain
- `OBJECTSTORE_SWIFT_SERVICE_NAME` (default: `swift`): Swift service name
- `OBJECTSTORE_SWIFT_REGION`: Swift endpoint region
- `OBJECTSTORE_SWIFT_CONTAINER_NAME`: Swift container (bucket) that Nextcloud should store the data in

Check the Nextcloud documentation for more information.

PHP Configuration

To customize PHP limits you can change the following variables:

- `PHP_MEMORY_LIMIT` (default `512M`) This sets the maximum amount of memory in bytes that a script is allowed to allocate. This is meant to help prevent poorly written scripts from eating up all available memory but it can prevent normal operation if set too tight.
- `PHP_UPLOAD_LIMIT` (default `512M`) This sets the upload limit (`post_max_size` and `upload_max_filesize`) for big files. Note that you may have to change other limits depending on your client, webserver or operating system. Check the Nextcloud documentation for more information.
- `PHP_OPCACHE_MEMORY_CONSUMPTION` (default `128`) This sets the `opcache.memory_consumption` value. It's the size of the shared memory storage used by OPcache, in megabytes.

Apache Configuration

To customize the Apache max file upload limit you can change the following variable:

- `APACHE_BODY_LIMIT` (default `1073741824` [1GiB]) This restricts the total size of the HTTP request body sent from the client. It specifies the number of *bytes* that are allowed in a request body. A value of **0** means **unlimited**.

Check the Nextcloud documentation for more information.

Using the image behind a reverse proxy and specifying the server host and protocol

By default, the apache image will replace the remote addr (IP address visible to Nextcloud) with the IP address from `X-Real-IP` if the request is coming from a reverse proxy in `10.0.0.0/8`, `172.16.0.0/12` or `192.168.0.0/16`. If you want Nextcloud to pick up the server host (`HTTP_X_FORWARDED_HOST`), protocol (`HTTP_X_FORWARDED_PROTO`) and client IP (`HTTP_X_FORWARDED_FOR`) from a trusted proxy, then disable rewrite IP and add the reverse proxy's IP address to `TRUSTED_PROXIES`.

- `APACHE_DISABLE_REWRITE_IP` (not set by default): Set to 1 to disable rewrite IP.
- `TRUSTED_PROXIES` (empty by default): A space-separated list of trusted proxies. CIDR notation is supported for IPv4.

If the `TRUSTED_PROXIES` approach does not work for you, try using fixed values for overwrite parameters.

- `OVERWRITEHOST` (empty by default): Set the hostname of the proxy. Can also specify a port.
- `OVERWRITEPROTOCOL` (empty by default): Set the protocol of the proxy, http or https.
- `OVERWRITECLIURL` (empty by default): Set the cli url of the proxy (e.g. https://mydnsname.example.com)
- `OVERWRITEWEBROOT` (empty by default): Set the absolute path of the proxy.
- `OVERWRITECONDADDR` (empty by default): Regex to overwrite the values dependent on the remote address.
- `FORWARDED_FOR_HEADERS` (empty by default): HTTP headers with the original client IP address

Check the Nexcloud documentation for more details.

Keep in mind that once set at install time, removing these environment variables later won't remove them from your `config/config.php`, due to how Nextcloud generates and merges the initial configuration at installation time. They can still, however, be removed manually from your `config/config.php`.

Handling `Warning: /var/www/html/config/$cfgFile differs from the latest version of this image at /usr/src/nextcloud/config/$cfgFile` (aka: Auto configuration and Nextcloud updates)

The image comes with special config files for Nextcloud that set parameters specific to containerized usage (e.g. `upgrade-disable-web.config.php`) or enable auto configuration via environment variables (e.g. `reverse-proxy.config.php`). Not keeping these files up-to-date when this warning appears may cause certain auto configuration environment variables to be ignored or the image to not work as documented or expected.

During a fresh Nextcloud installation, the latest version (from the image) of these files are copied into `/var/www/html/config` so that they are stored within your container's persistent storage and picked up by Nextcloud alongside your local configuration.

The copied files, however, are **not** automatically overwritten whenever you update your environment with a newer Nextcloud image. This is to prevent local changes in `/var/www/html/config` from being unexpectedly overwritten. This may lead to your image-specific configuration files becoming outdated and image functionality not matching that which is documented.

Within each image, the latest version of these config files are located in `/usr/src/nextcloud/config`.

A warning will be generated in the container log output when outdated image-specific configuration files are detected at startup in a running container. When you see this warning, you should manually compare (or copy) the files from `/usr/src/nextcloud/config` to `/var/www/html/config`. A command to copy these configs would e.g. be:

```
docker exec <container-name> sh -c "cp /usr/src/nextcloud/config/*.php /var/www/html/config"
```

As long as you have not modified any of the provided config files in `/var/www/html/config` (other than `config.php`) or only added new ones with names that do not conflict with the image specific ones, copying the new ones into place should be safe (but check the source path `/usr/src/nextcloud/config` for any newly named config files to avoid new overlaps just in case).

Auto configuration via hook folders

There are 5 hooks

- `pre-installation` Executed before the Nextcloud is installed/initiated
- `post-installation` Executed after the Nextcloud is installed/initiated
- `pre-upgrade` Executed before the Nextcloud is upgraded
- `post-upgrade` Executed after the Nextcloud is upgraded
- `before-starting` Executed before the Nextcloud starts

To use the hooks triggered by the `entrypoint` script, either

- Added your script(s) to the individual of the hook folder(s), which are located at the path `/docker-entrypoint-hooks.d` in the container
- Use volume(s) if you want to use script from the host system inside the container, see example.

Note: Only the script(s) located in a hook folder (not sub-folders), ending with `.sh` and marked as executable, will be executed.

Example: Mount using volumes

```
...
  app:
    image: nextcloud:stable

    volumes:
      - ./app-hooks/pre-installation:/docker-entrypoint-hooks.d/pre-installation
      - ./app-hooks/post-installation:/docker-entrypoint-hooks.d/post-installation
      - ./app-hooks/pre-upgrade:/docker-entrypoint-hooks.d/pre-upgrade
      - ./app-hooks/post-upgrade:/docker-entrypoint-hooks.d/post-upgrade
      - ./app-hooks/before-starting:/docker-entrypoint-hooks.d/before-starting
...
```

Running this image with `docker compose`

The easiest way to get a fully featured and functional setup is using a `compose.yaml` file. There are too many different possibilities to setup your system, so here are only some examples of what you have to look for.

At first, make sure you have chosen the right base image (fpm or apache) and added features you wanted (see below). In every case, you would want to add a database container and docker volumes to get easy access to your persistent data. When you want to have your server reachable from the internet, adding HTTPS-encryption is mandatory! See below for more information.

Base version - apache

This version will use the apache variant and add a MariaDB container. The volumes are set to keep your data persistent. This setup provides **no TLS encryption** and is intended to run behind a proxy.

Make sure to pass in values for `MYSQL_ROOT_PASSWORD` and `MYSQL_PASSWORD` variables before you run this setup.

```
services:
  # Note: MariaDB is external service. You can find more information about the configuration her
  # https://hub.docker.com/_/mariadb
  db:
    # Note: Check the recommend version here: https://docs.nextcloud.com/server/latest/admin_mar
    image: mariadb:lts
    restart: always
    command: --transaction-isolation=READ-COMMITTED
    volumes:
      - db:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=
```

```
      - MYSQL_PASSWORD=
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud

  # Note: Redis is an external service. You can find more information about the configuration he
  # https://hub.docker.com/_/redis
  redis:
    image: redis:alpine
    restart: always

  app:
    image: nextcloud
    restart: always
    ports:
      - 8080:80
    depends_on:
      - redis
      - db
    volumes:
      - nextcloud:/var/www/html
    environment:
      - MYSQL_PASSWORD=
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_HOST=db

volumes:
  nextcloud:
  db:
```

Then run `docker compose up -d`, now you can access Nextcloud at http://localhost:8080/ from your host system.

Base version - FPM

When using the FPM image, you need another container that acts as web server on port 80 and proxies the requests to the Nextcloud container. In this example a simple nginx container is combined with the Nextcloud-fpm image and a MariaDB database container. The data is stored in docker volumes. The nginx container also needs access to static files from your Nextcloud installation. It gets access to all the volumes mounted to Nextcloud via the `volumes` option. The configuration for nginx is stored in the configuration file `nginx.conf`, that is mounted into the container. An example can be found in the examples section here.

This setup provides **no TLS encryption** and is intended to run behind a reverse proxy.

Make sure to pass in values for `MYSQL_ROOT_PASSWORD` and `MYSQL_PASSWORD` variables before you run this setup.

```
services:
  # Note: MariaDB is an external service. You can find more information about the configuration
  # https://hub.docker.com/_/mariadb
```

```
  db:
    # Note: Check the recommend version here: https://docs.nextcloud.com/server/latest/admin_mar
    image: mariadb:lts
    restart: always
    command: --transaction-isolation=READ-COMMITTED
    volumes:
      - db:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=
      - MYSQL_PASSWORD=
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud

  # Note: Redis is an external service. You can find more information about the configuration he
  # https://hub.docker.com/_/redis
  redis:
    image: redis:alpine
    restart: always

  app:
    image: nextcloud:fpm
    restart: always
    depends_on:
      - redis
      - db
    volumes:
      - nextcloud:/var/www/html
    environment:
      - MYSQL_PASSWORD=
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_HOST=db

  # Note: Nginx is an external service. You can find more information about the configuration he
  # https://hub.docker.com/_/nginx/
  web:
    image: nginx:alpine-slim
    restart: always
    ports:
      - 8080:80
    depends_on:
      - app
    volumes:
      # https://docs.nextcloud.com/server/latest/admin_manual/installation/nginx.html
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    volumes_from:
      - app

volumes:
  nextcloud:
  db:
```

Then run `docker compose up -d`, now you can access Nextcloud at http://localhost:8080/ from your host system.

Docker Secrets

As an alternative to passing sensitive information via environment variables, `_FILE` may be appended to some the previously listed environment variables, causing the initialization script to load the values for those variables from files present in the container. In particular, this can be used to load passwords from Docker secrets stored in `/run/secrets/<secret_name>` files.

Currently, this is only supported for `NEXTCLOUD_ADMIN_PASSWORD`, `NEXTCLOUD_ADMIN_USER`, `MYSQL_DATABASE`, `MYSQL_PASSWORD`, `MYSQL_USER`, `POSTGRES_DB`, `POSTGRES_PASSWORD`, `POSTGRES_USER`, `REDIS_HOST_PASSWORD`, `SMTP_PASSWORD`, `OBJECTSTORE_S3_KEY`, and `OBJECTSTORE_S3_SECRET`.

If you set any group of `_FILE` based values (i.e. all of `MYSQL_DATABASE_FILE`, `MYSQL_USER_FILE`, `MYSQL_PASSWORD_FILE`), their non-`_FILE` counterparts will be ignored (`MYSQL_DATABASE`, `MYSQL_USER`, `MYSQL_PASSWORD`).

Any files containing secrets must be readable by the UID the container is running Nextcloud as (i.e. `www-data` / `33`).

Example:

```
services:
  # Note: PostgreSQL is external service. You can find more information about the configuration
  # https://hub.docker.com/_/postgres
  db:
    # Note: Check the recommend version here: https://docs.nextcloud.com/server/latest/admin_mar
    image: postgres:alpine
    restart: always
    volumes:
      - db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB_FILE=/run/secrets/postgres_db
      - POSTGRES_USER_FILE=/run/secrets/postgres_user
      - POSTGRES_PASSWORD_FILE=/run/secrets/postgres_password
    secrets:
      - postgres_db
      - postgres_password
      - postgres_user
  # Note: Redis is an external service. You can find more information about the configuration he
  # https://hub.docker.com/_/redis
  redis:
    image: redis:alpine
    restart: always

  app:
    image: nextcloud
    restart: always
    ports:
      - 8080:80
```

```
    volumes:
      - nextcloud:/var/www/html
    environment:
      - POSTGRES_HOST=db
      - POSTGRES_DB_FILE=/run/secrets/postgres_db
      - POSTGRES_USER_FILE=/run/secrets/postgres_user
      - POSTGRES_PASSWORD_FILE=/run/secrets/postgres_password
      - NEXTCLOUD_ADMIN_PASSWORD_FILE=/run/secrets/nextcloud_admin_password
      - NEXTCLOUD_ADMIN_USER_FILE=/run/secrets/nextcloud_admin_user
    depends_on:
      - redis
      - db
    secrets:
      - nextcloud_admin_password
      - nextcloud_admin_user
      - postgres_db
      - postgres_password
      - postgres_user

volumes:
  db:
  nextcloud:

secrets:
  nextcloud_admin_password:
    file: ./nextcloud_admin_password.txt # put admin password in this file
  nextcloud_admin_user:
    file: ./nextcloud_admin_user.txt # put admin username in this file
  postgres_db:
    file: ./postgres_db.txt # put postgresql db name in this file
  postgres_password:
    file: ./postgres_password.txt # put postgresql password in this file
  postgres_user:
    file: ./postgres_user.txt # put postgresql username in this file
```

Make your Nextcloud available from the internet

Until here, your Nextcloud is just available from your docker host. If you want your Nextcloud available from the internet adding SSL encryption is mandatory.

HTTPS - SSL encryption

There are many different possibilities to introduce encryption depending on your setup.

We recommend using a reverse proxy in front of your Nextcloud installation. Your Nextcloud will only be reachable through the proxy, which encrypts all traffic to the clients. You can mount your manually generated certificates to the proxy or use a fully automated solution which generates and renews the certificates for you.

In our examples section we have an example for a fully automated setup using a reverse proxy, a container for Let's Encrypt certificate handling, database and Nextcloud. It uses the popular nginx-proxy and acme-companion containers. Please check the according documentations before using this setup.

First use

When you first access your Nextcloud, the setup wizard will appear and ask you to choose an administrator account username, password and the database connection (unless of course you've provided all the necessary auto-config config values ahead of time).

For the database use `db` as host and `nextcloud` as table and user name. Also enter the password you chose in your `compose.yaml` file.

Update to a newer version

Updating the Nextcloud container is done by pulling the new image, throwing away the old container and starting the new one.

It is only possible to upgrade one major version at a time. For example, if you want to upgrade from version 14 to 16, you will have to upgrade from version 14 to 15, then from 15 to 16.

Since all data is stored in volumes, nothing gets lost. The startup script will check for the version in your volume and the installed docker version. If it finds a mismatch, it automatically starts the upgrade process. Don't forget to add all the volumes to your new container, so it works as expected.

```
$ docker pull nextcloud
$ docker stop <your_nextcloud_container>
$ docker rm <your_nextcloud_container>
$ docker run <OPTIONS> -d nextcloud
```

Beware that you have to run the same command with the options that you used to initially start your Nextcloud. That includes volumes, port mapping.

When using docker compose your compose file takes care of your configuration, so you just have to run:

```
$ docker compose pull
$ docker compose up -d
```

Adding Features

A lot of people want to use additional functionality inside their Nextcloud installation. If the image does not include the packages you need, you can easily build your own image on top of it. Start

your derived image with the `FROM` statement and add whatever you like.

```
FROM nextcloud:apache

RUN ...
```

The examples folder gives a few examples on how to add certain functionalities, like including the cron job, smb-support or imap-authentication.

If you use your own Dockerfile, you need to configure your docker compose file accordingly. Switch out the `image` option with `build`. You have to specify the path to your Dockerfile. (in the example it's in the same directory next to the `compose.yaml` file)

```
  app:
    build: .
    restart: always
    depends_on:
      - db
    volumes:
      - data:/var/www/html/data
      - config:/var/www/html/config
      - apps:/var/www/html/apps
```

If you intend to use another command to run the image, make sure that you set `NEXTCLOUD_UPDATE=1` in your Dockerfile. Otherwise the installation and update will not work.

```
FROM nextcloud:apache

...

ENV NEXTCLOUD_UPDATE=1

CMD ["/usr/bin/supervisord"]
```

Updating your own derived image is also very simple. When a new version of the Nextcloud image is available run:

```
docker build -t your-name --pull .
docker run -d your-name
```

or for docker compose:

```
docker compose build --pull
docker compose up -d
```

The `--pull` option tells docker to look for new versions of the base image. Then the build instructions inside your `Dockerfile` are run on top of the new image.

Migrating an existing installation

You're already using Nextcloud and want to switch to docker? Great! Here are some things to look out for:

1. Define your whole Nextcloud infrastructure in a `compose.yaml` file and run it with `docker compose up -d` to get the base installation, volumes and database. Work from there.
2. Restore your database from a mysqldump (db is the name of your database container / service name)
   - To import from a MySQL dump use the following commands

```
docker compose cp ./database.dmp db:/dmp
docker compose exec db sh -c "mysql --user USER --password PASSWORD nextcloud < /dmp"
docker compose exec db rm /dmp
```

   - To import from a PostgreSQL dump use to following commands

```
docker compose cp ./database.dmp db:/dmp
docker compose exec db sh -c "psql -U USER --set ON_ERROR_STOP=on nextcloud < /dmp"
docker compose exec db rm /dmp
```

3. Edit your config.php
   1. Set database connection
      - In case of MySQL database

```
'dbhost' => 'db:3306',
```

      - In case of PostgreSQL database

```
'dbhost' => 'db:5432',
```

   2. Make sure you have no configuration for the `apps_paths`. Delete lines like these

```
'apps_paths' => array (


    ),
),
```

   3. Make sure to have the `apps` directory non writable and the `custom_apps` directory writable

```
'apps_paths' => array (
0 =>


  ),
1 =>
'path'


  ),
),
```

4. Make sure your data directory is set to /var/www/html/data

```
'datadirectory' => '/var/www/html/data',
```

4. Copy your data (`app` is the name of your Nextcloud container / service name):

```
docker compose cp ./data/ app:/var/www/html/
docker compose exec app chown -R www-data:www-data /var/www/html/data
docker compose cp ./theming/ app:/var/www/html/
docker compose exec app chown -R www-data:www-data /var/www/html/theming
docker compose cp ./config/config.php app:/var/www/html/config
docker compose exec app chown -R www-data:www-data /var/www/html/config
```

   If you want to preserve the metadata of your files like timestamps, copy the data directly on the host to the named volume using plain `cp` like this:

```
cp --preserve --recursive ./data/ /path/to/nextcloudVolume/data
```

5. Copy only the custom apps you use (or simply redownload them from the web interface):

```
docker compose cp ./custom_apps/ app:/var/www/html/
docker compose exec app chown -R www-data:www-data /var/www/html/custom_apps
```

Migrating from a non-Alpine image to an Alpine image

If you already use one of our non-Alpine images, but want to switch to an Alpine-based image, you may experience permissions problems with your existing volumes. This is because the Alpine images uses a different user ID for `www-data`. So, you must change the ownership of the `/var/www/html` (or `$NEXTCLOUD_DATA_DIR`) folder to be compatible with Alpine:

```
docker exec container-name chown -R www-data:root /var/www/html
```

After changing the permissions, restart the container and the permission errors should disappear.

Reporting bugs or suggesting enhancements

If you believe you've found a bug in the image itself (or have an enhancement idea specific to the image), please search for already reported bugs and enhancement ideas.

If there is a relevant existing open issue, you can either add to the discussion there or upvote it to indicate you're impacted by (or interested in) the same issue.

If you believe you've found a new bug, please create a new Issue so that others can try to reproduce it and remediation can be tracked.

GitHub Issues or Pull Requests GitHub Issues or Pull Requests by label
GitHub Issues or Pull Requests by label GitHub Issues or Pull Requests by label

If you have any problems or usage questions while using the image, please ask for assistance on the Nextcloud Community Help Forum rather than reporting them as "bugs"

(unless they really are bugs of course). This helps the maintainers (who are volunteers) remain focused on making the image better (rather than responding solely to one-on-one support issues). (Tip: Some of the maintainers are also regular responders to help requests on the Nextcloud Community Help Forum.)

Discourse Users Discourse Posts

Most Nextcloud Server matters are covered in the official Nextcloud Admin Manual or the other official Nextcloud documentation (which are routinely updated).

Nextcloud LDAP authentication

Link:
https://docs.nextcloud.com/server/stable/admin_manual/configuration_user/user_auth_ldap.html#

User authentication with LDAP

Nextcloud ships with an LDAP application to allow LDAP users (including Active Directory) to appear in your Nextcloud user listings. These users will authenticate to Nextcloud with their LDAP credentials, so you don’t have to create separate Nextcloud user accounts for them. You will manage their Nextcloud group memberships, quotas, and sharing permissions just like any other Nextcloud user.

Note

The PHP LDAP module is required; this is supplied by `php-ldap` on most distributions.

The LDAP application supports:

- LDAP group support
- File sharing with Nextcloud users and groups
- Access via WebDAV and Nextcloud Desktop Client
- Versioning, external Storage and all other Nextcloud 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)
- Optional: Allow users to change their LDAP password from Nextcloud

Note

A non-blocking or correctly configured SELinux setup is needed for the LDAP backend to work. Please refer to the SELinux configuration.

Configuration

First enable the `LDAP user and group backend` app on the Apps page in Nextcloud. 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.

Note

Do not configure any failover LDAP hosts here. See Advanced settings for instructions instead.

At a minimum you must supply the LDAP server’s hostname. If your server requires authentication, enter your credentials on this tab. Nextcloud will then attempt to auto-detect the server’s port and base DN. The base DN and port are mandatory, so if Nextcloud cannot detect them you must enter them manually.

LDAP wizard, server tab

Server configuration:

Configure one or more LDAP servers. Click the **Delete Configuration** button to remove the active configuration.

Host:

The host name or IP address of the LDAP server. It can also be a **ldaps://** URI. If you enter the port number, it speeds up server detection.

Examples:

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

Port:

The port on which to connect to the LDAP server. The field is disabled in the beginning of a new configuration. If the LDAP server is running on a standard port, the port will be detected automatically. If you are using a non-standard port, Nextcloud 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=nextcloudsystemuser,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. Nextcloud attempts to determine the Base DN according to the provided User DN or the provided Host, and you must enter it manually if Nextcloud does not detect it.

Example:

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

Users tab

Use this to control which LDAP users are listed as Nextcloud users on your Nextcloud server. In order to control which LDAP users can login to your Nextcloud server use the **Login Attributes** tab. 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.

User filter

Only those object classes:

Nextcloud will determine the object classes that are typically available for user objects in your LDAP. Nextcloud 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 Nextcloud. 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 LDAP Query:

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

```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com))
```

x users found:

This is an indicator that tells you approximately how many users will be listed in Nextcloud. The number updates automatically after any changes.

Login attributes tab

The settings in the Login Attributes tab determine which LDAP users can log in to your Nextcloud 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 Users tab by using a raw LDAP filter.

Login filter

LDAP Username:

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

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 LDAP Query:

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=nextcloudusers,ou=groups,
dc=example,dc=com)(uid=%uid)
```

- username or email address:

```
((&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com)(|(uid=%uid)(mail=%uid)))
```

Groups tab

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

Group filter

Only these object classes:

Nextcloud will determine the object classes that are typically available for group objects in your LDAP server. Nextcloud 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 these groups:

Nextcloud will generate a list of available groups found in your LDAP server. Then you select the group or groups that get access to your Nextcloud server.

Edit LDAP Query:

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 Nextcloud. The number updates automatically after any change.

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 four parts:

- Connection Settings
- Directory Settings
- Special Attributes
- User Profile Attributes

Connection settings

Advanced settings

Configuration Active:

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

Backup (Replica) Host:

If you have a backup LDAP server, enter the connection settings here. Nextcloud 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 Nextcloud 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! *Note*: The effect of this setting depends on the PHP system configuration. It does for example not work with the [official Nextcloud container image](https://github.com/nextcloud/docker). To disable certificate verification for a particular use, append the following configuration line to your */etc/ldap/ldap.conf*:

```
TLS_REQCERT                                                    ALLOW
```

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

Directory settings.

User Display Name Field:

The attribute that should be used as display name in Nextcloud.

- Example: *displayName*

2nd User Display Name Field:

An optional second attribute displayed in brackets after the display name, for example using the `mail` att `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:
  displayName
  mail

Disable users missing from LDAP

If this is enabled, users which are missing from LDAP, also known as remnants, will behave as if disabled in Nextcloud. This means for instance that public shares by these users will not work anymore. see also LDAP user cleanup.

Group Display Name Field:

The attribute that should be used as Nextcloud group name. Nextcloud 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:
  cn
  description

Group Member association:

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

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

- Example: *uniquemember*

Nested groups:

Enable group member retrieval from sub groups.

To allow user listing and login from nested groups, please see **User listing and login per nested groups** in the section **Troubleshooting, Tips and Tricks**.

Enable LDAP password changes per user:

Allow LDAP users to change their password and allow Super Administrators and Group Administrators to change the password of their LDAP users.

To enable this feature, the following requirements have to be met:

- General requirements:

> - Access control policies must be configured on the LDAP server to grant permissions for password changes. The User DN as configured in *Server Settings* needs to have write permissions in order to update the userPassword attribute.
> - Passwords are sent in plaintext to the LDAP server. Therefore, transport encryption must be used for the communication between Nextcloud and the LDAP server, e.g. employ LDAPS.
> - Enabling password hashing on the LDAP server is highly recommended. While Active Directory stores passwords in a one-way format by default, OpenLDAP users could configure the `ppolicy_hash_cleartext` directive of the ppolicy overlay that ships with OpenLDAP.

- Additional requirements for Active Directory:

> - At least a 128-bit transport encryption must be used for the communication between Nextcloud and the LDAP server.
> - Make sure that th `fUserPwdSupport` char of the dSHeuristics is configured to employ th `userPassword` attribute `unicodePwd` alias. While this is set accordingly on AD LDS by default, this is

> not the case on AD DS.

Default password policy DN:

This feature requires OpenLDAP with ppolicy. The DN of a default password policy will be used for password expiry handling in the absence of any user specific password policy. Password expiry handling features the following:

- When a LDAP password is about to expire, display a warning message to the user showing the number of days left before it expires. Password expiry warnings are displayed through the notifications app for Nextcloud.
- Prompt LDAP users with expired passwords to reset their password during login, provided that an adequate number of grace logins is still available.

Leave the setting empty to keep password expiry handling disabled.

For the password expiry handling feature to work, LDAP password changes per user must be enabled and the LDAP server must be running OpenLDAP with its ppolicy module configured accordingly.

- Example: *cn=default,ou=policies,dc=my-company,dc=com*

Special attributes

Special Attributes.

Quota Field:

Nextcloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, and it will return human-readable values, e.g. “2 GB”.

- Example: *NextcloudQuota*

Warning

LDAP quota parameters override quota parameters set in the Nextcloud user management page.

Quota Default:

Specifies a default quota for LDAP users who do not have a quota set in the above Quota Field.

- Example: *15 GB*

Warning

LDAP quota parameters override quota parameters set in the Nextcloud user management page.

Email Field:

Set the user's email from their LDAP attribute. Leave it empty for default behavior.

- Example: *mail*

User Home Folder Naming Rule:

By default, the Nextcloud server creates the user directory in your Nextcloud data directory and gives it the Nextcloud username, .e.g. `/var/www/nextcloud/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 Nextcloud installations 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 migrated Nextcloud installations the old behavior still applies, which is using the Nextcloud username as the home folder when an LDAP attribute is not set. You may change this enforcing the home folder rule with the `occ` command in Nextcloud, like this example on Ubuntu:

```
sudo -E -u www-data php occ config:app:set user_ldap enforce_home_folder_naming_rule --value=1
```

User Profile attributes

User Profile Attributes.

After configuring those attributes, the User Profile data will be overwritten with the according data from LDAP. The checksum of data from LDAP will be stored in us ngs `user_ldap`, `lastProfileChecksum` and profile update is skipped as long as data from LDAP doesn't change. If `memcache.distributed` is enabled in `config.php` the checksum will be cached and the checking will be skipped, as long as the cached value exists (expires after `ldapCacheTTL` seconds).

Please be aware:

- The user can change the data in profile, but it will get overwritten if changed in LDAP
- The user can change the visibility scope in profile
- The default visibility can be adjusted with setting the `account_manager.default_property_scope` array in `config.php`
- If multiple attribute values are present, only the first distributed value is used

- All user profile properties are limited to 2048 character
- Having misformatted data in LDAP will most probably leave you with empty user profile fields
- Setting the global `profile.enabled => false` on `config.php` skips the code

By calling `sudo -E -u www-data php occ ldap:check-user --update <uid>` the users data from LDAP will be displayed and the profile gets updated. To get the correct `<uid>` value for any user you can use `php occ user:list`.

Note

After unsetting an attribute name here, the data won't be deleted from user profile. Setting an nonexisting attribute will empty the corresponding profile field.

Phone Field:

The LDAP Attribute holding the phone number, to copy to the Profile Phone field. The phone number has to be formatted in international syntax without delimiters (E.164). Be sure to format phone numbers like `+4966612345678`.

- Example: *telephoneNumber*
- Example: *mobile*

Note

You should set your `default_phone_region` in `config.php`.

Website Field:

The LDAP attribute holding the website URI. The URI must start with `https://` or `http://` others are currently not allowed in Nextcloud user profile. If us `labeledURI` attributes the label (everything after first SPACE) gets removed.

- Example: *wWWHomePage*
- Example: *labeledURI*

Address Field:

The LDAP attribute holding the users address. Named Location on user profile page. Nextcloud wants a single line value like `city, country` or `somewhere under the loving sun`. Multi line postalAddress format will get reformatted, DOLLAR sign delimiter gets replaced with COMMA+SPACE.

- Example: *postalAddress*
- Example: *localityName*

Twitter Field:

The LDAP attribute holding the Twitter account name.

Fediverse Field:

The LDAP attribute holding the users Fediverse address.

Organisation Field:

The LDAP attribute holding the Organisation name.

- Example: *company*
- Example: *o* or *organizationName*

Role Field:

The LDAP attribute holding the organizational role, within the organisation or job title.

- Example: *title*

Headline Field:

The LDAP attribute holding the users headline.

Biography Field:

The LDAP attribute holding the users about i.e. short biography. Multi line value with unix LF line ending. Windows CRLF and Macintosh CR line endings will be replaced with unix LF line ending.

Birthdate Field:

The LDAP attribute holding the user's date of birth. Allowed formats:

- LDAP GeneralizedTime
- `YYYY-MM-DD`
- `YYYYMMDD`

Expert settings

Expert settings.

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 Nextcloud 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 Nextcloud, i.e. that it is checking all other activated user backends (including local Nextcloud users). On collisions a random number (between 1000 and 9999) will be attached to the retrieved value. For example, if “alice” exists, the next username may be “alice_1337”.

The internal username is the default name for the user home folder in Nextcloud. 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 behavior. Changes will affect only newly mapped LDAP users.

When configuring this, be aware that the username in Nextcloud is considered immutable and cannot be changed afterwards. This can cause issues when using an attribute that might change, e.g. the email address of a user that will get changed during name change.

- Example: *uid*

Override UUID detection

By default, Nextcloud 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 behavior. 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 Nextcloud in production use and clearing the bindings (see the `User and Group Mapping` section below).

- Example: *cn*

Username-LDAP User Mapping

Nextcloud 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 Nextcloud. This requires a mapping from Nextcloud 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 Nextcloud by checking the UUID value.

The same is valid for groups.

The internal Nextcloud name is used all over in Nextcloud. Clearing the Mappings will have leftovers everywhere. Never clear the mappings in a production environment, but only in a testing or experimental server.

Warning

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

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, Nextcloud will try to bind to the Nextcloud 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.

Additional configuration options via occ

Few configuration settings can only be set on command line via `occ`.

Attribute update interval

The LDAP backend will update user information that is used within Nextcloud with the values provided by the LDAP server. For instance these are email, quota or the avatar. This happens on every login, the first detection of a user from LDAP and regularly by a background job.

The interval value determines the time between updates of the values and is used to avoid frequent overhead, including time-expensive write actions to the database.

The interval is described in seconds and it defaults to 86400 equalling a day. It is not a per-configuration option.

The value can be modified by:

```
sudo -E -u www-data php occ config:app:set user_ldap updateAttributesInterval --value=86400
```

A value of 0 will update it on every of the named occasions.

Administrative Group mapping

It is possible to promote **one** LDAP per connection as an admin group, so that all its members also have administrative privileges in Nextcloud.

A group can either be promoted via a dedicated `occ` call providing a group parameter that can be either a nextcloud group ID or a group name that will be search against. When a search is executed an exact match is required.

Example usage:

```
$ sudo -E -u www-data php occ ldap:promote-group --help
Description:
  declares the specified group as admin group (only one is possible per LDAP configuration)

Usage:
                                ldap:promote-group                [options]

Arguments:
        group                                                                the

Options:
               -y,        --yes
…

#                                                                          Example
$   sudo   -E   -u   www-data   php   occ   ldap:promote-group       "Nextcloud   Admi
Promote     Nextcloud     Admins     to     the     admin     group     (y|N)?     y
Group              Nextcloud              Admins              was              promoted

$   sudo   -E   -u   www-data   php   occ   ldap:promote-group       "Paramount   Cou
PromoteNextcloudAdminstotheadmingroupanddemoteNextcloudAdmins(GroupID:nextcloud_admins)(y|N)?y
Group              Paramount              Court              was              promoted

$   sudo   -E   -u   www-data   php   occ   ldap:promote-group       "Paramount   Cou
The          specified          group          is          already          promoted
```

Note

Note the group ID will only be displayed when it differs from the group's display name.

It is also possible to set the admin group mappi`g occ ldap:set-config $configId ldapAdminGroup $groupId`, but as the Nextcloud group ID might not be known (yet) it is recommended (especially for automated setups) to use the *promote-group* command, that would also pull in the group and determine the group ID.

In order to demote or reset a promotion, an empty string should be set against to the targeted config's ldapAdminGroup:

```
#       Reset       an       admin       group       mapping       via       set-config
occ           ldap:set-config           $configId           ldapAdminGroup           ""
#                                                                          Example
```

```
occ ldap:set-config s01 ldapAdminGroup ""
```

Tip

To have more than one administrative groups in a connection, create a holding group in your LDAP directory that contains the single groups as nested members, and promote this one.

Nextcloud avatar integration

Nextcloud 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*.

Profile picture fetched from LDAP.

If the *jpegPhoto* or *thumbnailPhoto* attribute is not set or empty, then users can upload and manage their avatars on their Nextcloud Personal pages. Avatars managed in Nextcloud 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 Nextcloud. LDAP avatars override Nextcloud avatars, and when an LDAP avatar is deleted then the most recent Nextcloud avatar replaces it.

Photos served from LDAP are automatically cropped and resized in Nextcloud. This affects only the presentation, and the original image is not changed.

Use a specific attribute or turn off loading of images

It is possible to turn off the avatar integration or specify a single, different attribute to read the image from. It is expected to contain image data just like *jpegPhoto* or *thumbnailPhoto* do.

The behaviour can be changed using the occ command line tool only. Essentially those options are available:

- The default behaviour as described above should be used
  `occ ldap:set-config "s01" "ldapUserAvatarRule" "default"`
- User images shall not be fetched from LDAP
  `occ ldap:set-config "s01" "ldapUserAvatarRule" "none"`
- The image should be read from the attribute “selfiePhoto”
  `occ ldap:set-config "s01" "ldapUserAvatarRule" "data:selfiePhoto"`

The “s01” refers to the configuration ID as can be retrieved per `occ ldap:show-config`.

Troubleshooting, tips and tricks

Logging

Nextcloud's LDAP implementation is capable of logging lots of additional details about its activities. When diagnosing problems, it can be useful to temporarily adjust y `loglevel` to II FO (`1`) or DEBUG (`0`).

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 Nextcloud 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 Nextcloud versions, no further tweaks need to be done to make Nextcloud work with Active Directory. Nextcloud will automatically find the correct configuration in the set-up process.

memberOf / read memberof permissions

If you want to us `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.

User listing and login per nested groups

When it is intended to allow user listing and login based on a specific group having subgroups ("nested groups"), checking **Nested groups** on **Directory Settings** is not enough. Also the User (and Login) filter need to be changed, by specifying the `LDAP_MATCHING_RULE_IN_CHAIN` matching rule. Change the filter parts containing the *memberof* condition according to this example:

> - (memberof=cn=Nextcloud Users Group,ou=Groups,…)

to

> - (memberof:1.2.840.113556.1.4.1941:=cn=Nextcloud Users Group,ou=Groups,…)

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.

Nextcloud LDAP internals

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

User and group mapping

In Nextcloud 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.

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.

The attributes of users are fetched on demand (i.e. for sharing autocompletion or in the user management) and then stored inside the Nextcloud database to allow a better performance on our side. They are typically checked twice a day in batches from all users again. Beside that they are also refreshed during a login for this user or can be fetched manually via the occ comma `occ ldap:check-user --update USERID` where `USERID` is Nextcloud's user id.

For groups, a cache of memberships is stored in the database to be able to trigger events when a membership is added or removed. This cache is updated by a background job, and can be force updated using `occ ldap:check-group --update GROUPID`.

Caching

The LDAP information is cached in Nextcloud memory cache, and you must install and configure the memory cache (see Memory caching). The Nextcloud **Cache** helps to speed up user interactions and sharing. It is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

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 a Nextcloud-Server and LDAP-Server, **Paged Results** must be supported.

Nextcloud 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.

Handling with backup server

When Nextcloud is not able to contact the main LDAP server, Nextcloud 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 Nextcloud will connect to it instead. When you have scheduled downtime, check **Disable Main Server** to avoid unnecessary connection attempts.

Note

When a LDAP object’s name or surname, that is display name attribute, by default “displayname”, is left empty, Nextcloud will treat it as an empty object, therefore no results from this user or AD-Object will be shown to avoid gathering of technical accounts.

Remove Nextcloud Share from Database

Link: https://blog.cubieserver.de/2018/remove-nextcloud-share-from-database/

Posted on Mar 11, 2018

#nextcloud #share #mysql #database

My Nextcloud instance is currently suuuuper slow, because one of the federated Nextcloud instances that *has shared a file with me* is offline. Apparently, this issue has been addressed a long time ago, but I'm still having this problem.

When I try to "unshare" the file in the Web GUI, I simply get (after a looong wait) the error message `Error deleting file "xyz".`. The HTTP DELETE operation sent to the backend returns with a 503 Service Unavailable, as well as some WebDAV exception in XML.

Because I got quite annoyed by this (it outright made the Web interface unusable), I decided to take it into my own hands and forcefully remove the entry from the database. Easier said than done, as Nextcloud's database and table structure is quite complex.

```
MariaDB [nextcloud]> show tables;
+-------------------------------+
| Tables_in_nextcloud           |
+-------------------------------+
| oc_accounts                   |
| oc_activity                   |
| oc_activity_mq                |
| oc_addressbookchanges         |
| oc_addressbooks               |
| oc_admin_sections             |
| oc_admin_settings             |
| oc_appconfig                  |
| oc_audioplayer_albums         |
| oc_audioplayer_artists        |
| oc_audioplayer_genre          |
| oc_audioplayer_playlist_tracks |
| oc_audioplayer_playlists      |
| oc_audioplayer_stats          |
```

```
| oc_audioplayer_streams         |
| oc_audioplayer_tracks          |
| oc_authtoken                   |
| oc_bookmarks                   |
| oc_bookmarks_tags              |
| oc_bruteforce_attempts         |
| oc_calendarchanges             |
| oc_calendarobjects             |
| oc_calendarobjects_props       |
| oc_calendars                   |
| oc_calendarsubscriptions       |
| oc_cards                       |
| oc_cards_properties            |
| oc_comments                    |
| oc_comments_read_markers       |
| oc_credentials                 |
| oc_dav_shares                  |
| oc_documents_invite            |
| oc_documents_member            |
| oc_documents_op                |
| oc_documents_revisions         |
| oc_documents_session           |
| oc_external_applicable         |
| oc_external_config             |
| oc_external_mounts             |
| oc_external_options            |
| oc_federated_reshares          |
| oc_file_locks                  |
| oc_filecache                   |
| oc_files_trash                 |
| oc_flow_checks                 |
| oc_flow_operations             |
| oc_group_admin                 |
| oc_group_user                  |
| oc_groups                      |
| oc_jobs                        |
| oc_ldap_group_mapping          |
| oc_ldap_group_members          |
| oc_ldap_user_mapping           |
| oc_mimetypes                   |
```

```
| oc_mounts                     |
| oc_news_feeds                 |
| oc_news_folders               |
| oc_news_items                 |
| oc_notes_meta                 |
| oc_notifications              |
| oc_oauth2_access_tokens       |
| oc_oauth2_clients             |
| oc_podcasts_episodes          |
| oc_podcasts_feeds             |
| oc_polls_comments             |
| oc_polls_dts                  |
| oc_polls_events               |
| oc_polls_notif                |
| oc_polls_particip             |
| oc_polls_particip_text        |
| oc_polls_txts                 |
| oc_preferences                |
| oc_privatedata                |
| oc_properties                 |
| oc_schedulingobjects          |
| oc_share                      |
| oc_share_external             |
| oc_storages                   |
| oc_systemtag                  |
| oc_systemtag_group            |
| oc_systemtag_object_mapping   |
| oc_trusted_servers            |
| oc_twofactor_backupcodes      |
| oc_users                      |
| oc_vcategory                  |
| oc_vcategory_to_object        |
+-------------------------------+
86 rows in set (0.00 sec)
```

I’m using Nextcloud 12.0.5 with MariaDB 10.1.

So log into your MySQL instance, select the appropriate database and drop tables – after backing up your database, of course!

There seems to be something interesting in table `oc_mounts`:

```
MariaDB [nextcloud]> select * from oc_mounts;
+----+------------+---------+---------+--------------------------+----------+
| id | storage_id | root_id | user_id | mount_point              | mount_id |
+----+------------+---------+---------+--------------------------+----------+
| .p.. more entries here ...                                                  |
| 36 |         33 |    6597 | user1   | /user1/files/filexyz/    |     NULL |
+----+------------+---------+---------+--------------------------+----------+
```

But also `oc_share_external` shows a hit:

```
MariaDB [nextcloud]> select * from oc_share_external;
+----+------------------------------+-----------+-----------------+----------+----------
+---------+-------+-----------+----------------------------------+----------+
| id | remote                       | remote_id | share_token     | password | name     |
owner   | user  | mountpoint | mountpoint_hash                  | accepted |
+----+------------------------------+-----------+-----------------+----------+----------
+---------+-------+-----------+----------------------------------+----------+
|  6 | https://remote-nextcloud.com/ |         5 | jasdlk49wJSD92A |          | /filexyz |
user1   | user2 | /filexyz  | 47f19b20f09d33e4abc4166d611e35b7 |        1 |
+----+------------------------------+-----------+-----------------+----------+----------
+---------+-------+-----------+----------------------------------+----------+
```

So let’s drop those rows and see what happens:

```
1
2
```

```
DELETE FROM oc_share_external WHERE id = 6
LIMIT 1;
Query OK, 1 row affected (0.00 sec)
```

```
1
2
```

```
DELETE FROM oc_mounts WHERE id = 36 LIMIT
1;
Query OK, 1 row affected (0.00 sec)
```

Quick test `Ctrl-Shift-R` – everything is plenty fast again!

Author Avatar
Author: Jack Henschel
Contact me
Cloud computing engineer, IT security specialist, avid cyclist.