# Nextcloud Docker

# Mettre en place Nextcloud Docker

## Prérequis :

- Avoir une base de donnée `Postgres` déployé en local
  - Voir déploiement Postgres
- Avoir un nom de domaine
  - Dans notre cas le nom de domaine utilisé est : `miraceti.net`

Il existe plusieurs manière de mettre en place `Nextcloud` Docker. Dans notre cas nous allons utiliser une base de données `Postgres` déjà déployé avec Docker

## Configuration nom de domaine

Grâce à notre nom de domaine, nous allons pouvoir créer des sous domaine (en CNAME `A`).

Sur le site de nom de domaine respectif, dans la section DNS, ajouter l'entrée suivante :

- cloud.miraceti.net A ip.adress.server

## Configuration Docker Compose

> Pour accéder à notre base de données externe, nous devons importer le `network` lié à la base de donnée (dans notre cas le network est "`bddnetwork`").
> De plus dans la variable d'environnement `POSTGRES_HOST` du conteneur de nextcloud **on doit renseigner le nom du conteneur BDD externe** (Dans notre cas "`postgres`")

Fichier `docker-compose.yaml` :

```
services:

  nextcloud:
    image: ${NEXTCLOUD_IMAGE_TAG}
    container_name: nextcloud
    networks:
```

```
      - bddnetwork
    ports:
      - 8082:80
    volumes:
      - /path/to/data/nextcloud:/var/www/html
      - /path/to/conf/nextcloud/config:/var/www/html/config
    restart: unless-stopped
    environment:
      TZ: ${NEXTCLOUD_TIMEZONE}
      POSTGRES_HOST: postgres
      DB_PORT: 5432
      POSTGRES_DB: ${NEXTCLOUD_DB_NAME}
      POSTGRES_USER: ${NEXTCLOUD_DB_USER}
      POSTGRES_PASSWORD: ${NEXTCLOUD_DB_PASSWORD}
      REDIS_HOST: redis-nextcloud
      REDIS_HOST_PORT: 6379
      REDIS_HOST_PASSWORD: ${NEXTCLOUD_REDIS_PASSWORD}
#      TRUSTED_PROXIES: ${NEXTCLOUD_HOSTNAME}
      OVERWRITECLIURL: ${NEXTCLOUD_URL}
      OVERWRITEPROTOCOL: https
      OVERWRITEHOST: ${NEXTCLOUD_HOSTNAME}
#      PHP_MEMORY_LIMIT: 2048M□□□# If you want to overwritte Php Memory Limit
#      PHP_UPLOAD_LIMIT: 2048M□□□# If you want to overwritte Php Upload Limit

  redis-nextcloud:
    image: ${NEXTCLOUD_REDIS_IMAGE_TAG}
    container_name: redis-nextcloud
    command: ["redis-server", "--requirepass", "$NEXTCLOUD_REDIS_PASSWORD"]
    volumes:
      - /path/to/data/redis-nextcloud:/data
    networks:
      - bddnetwork
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 5s
      retries: 3
      start_period: 60s
    restart: unless-stopped
```

```
volumes:
 nextcloud:
 redis-nextcloud:

networks:
  bddnetwork:
    external: true
```

Fichier `.env` :

```
# Nextcloud Variables
NEXTCLOUD_REDIS_IMAGE_TAG=redis:7.2
NEXTCLOUD_IMAGE_TAG=nextcloud:29.0
NEXTCLOUD_REDIS_PASSWORD=5454fdsLJDSqd45
NEXTCLOUD_DB_NAME=nextclouddb
NEXTCLOUD_DB_USER=nextclouddbuser
NEXTCLOUD_DB_PASSWORD=q5s4dazelqksdjfga
NEXTCLOUD_URL=https://cloud.miraceti.net
NEXTCLOUD_HOSTNAME=cloud.miraceti.net
# Timezone inside container
# A list of these tz database names can be looked up at Wikipedia
# https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
NEXTCLOUD_TIMEZONE=Europe/Paris
```

> Démarrer le docker compose devrait rendre Nextcloud accessible en local, ou sur l'adresse IP de votre serveur sur le réseau local, sur le port `8082`.

## Configuration Apache2

Création d'un nouveau host dans `/etc/apache2/sites-available` avec le nom suivant "`cloud.miraceti.net.conf`":

```
<VirtualHost *:80>
        ServerName cloud.miraceti.net
```

```
□ErrorLog ${APACHE_LOG_DIR}/cloud.miraceti.net.log
        CustomLog ${APACHE_LOG_DIR}/cloud.miraceti.net.log combined

□ProxyPreserveHost On
□ProxyRequests Off
        ProxyPass / http://localhost:8082/□□□# Port Used by container Nextcloud
        ProxyPassReverse / http://localhost:8082/□# Port Used by container Nextcloud

        <IfModule mod_dav.c>
           Dav off
        </IfModule>

□  <Proxy *>
          Order deny,allow
          Allow from all
        </Proxy>
</VirtualHost>
```

Vérification de la nouvelle configuration :

```
apachectl configtest
```

Activation de la nouvelle configuration :

```
sudo a2ensite cloud.miraceti.net.conf
```

# Utiliser la console Nextcloud "occ"

# L'utilisateur Nextcloud’s command-line

> La commande occ de Nextcloud (origine de "ownCloud Console") est l'interface de ligne de commande de Nextcloud. Vous pouvez effectuer de nombreuses opérations courantes sur le serveur avec occ, comme l'installation et la mise à niveau de Nextcloud, la gestion des utilisateurs, le scan de fichiers, du cryptage, des mots de passe, des paramètres LDAP, etc...

## Trouver le bon utilisateur

Pour utiliser l'interface ''occ'' il faut être préalablement connecté avec l'utilisateur disposant des droits. Pour connaître l'utilisateur qui possède les droits il faut se connecter dans le conteneur.

### Se connecter dans le conteneur

```
sudo docker exec -it nextcloud bash
```

### Trouver l'interface occ

Une fois dans le conteneur, rechercher l'emplacement du fichier ''occ''. Il est généralement à l'emplacement suivant : `/var/www/nextcloud/occ` dans notre cas il est dans `/var/www/html/occ` .

Pour exécuter le script ''occ'' il faut utiliser `php` par conséquent il faudra effectuer la commande suivante :

```
root@980bf10821234:/var/www/html# php occ
Console has to be executed with the user that owns the file config/config.php
Current user id: 0
Owner id of config.php: 33
Try adding 'sudo -u #33' to the beginning of the command (without the single quotes)
If running with 'docker exec' try adding the option '-u 33' to the docker command (without the single quotes)
```

La commande nous retourne une erreur pour nous informer que nous essayons d'exécuter le script avec le mauvais utilisateur. Elle nous informe aussi que le bon utilisateur est l'utilisateur 33.

Pour lister les utilisateurs existants dans le conteneur on peux utiliser la même commande que pour un Linux.

`cat /etc/passwd` Ce qui nous retourne les lignes suivantes :

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
..
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
...
```

Dans notre cas l'utilisateur 33 est `www-data`. Maintenant que nous connaissons le bon utilisateur, nous devons sortir de conteneur et se reconnecter en tant que l'utilisateur `www-data`.

# Se connecter en tant que

Pour se connecter ***en tant que*** dans un conteneur, il suffit de rajouter l'option `-u user_id` se qui donne la commande suivante :

```
sudo docker exec -it -u 33 nextcloud bash
```

## Utiliser occ

Maintenant que nous sommes connecté avec le bon utilisateur nous pouvons exécuter l'invite de commande Nextcloud.

```
www-data@980bf1081234:~/html$ php occ
Nextcloud 23.0.0

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
```

```
  -n, --no-interaction  Do not ask any interactive question
      --no-warnings     Skip global warnings, show command output only
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output, 2 for more
verbose output and 3 for debug

Available commands:
...
```

# Le scan de fichier

## Scanner des nouveaux fichiers

Il peut être intéressant de transférer des fichiers d'un serveur à l'autre directement dans l'emplacement de stockage de Nextcloud sans passer par l'interface d'upload. Dans ce cas il faudra réaliser un scan de fichiers pour les voir apparaître sur Nextcloud une fois le transfert terminé.

### Se connecter en tant que

Pour réaliser le scan de fichier il faut être connecté avec l'utilisateur disposant des droits sur l'invite de commande Nextcloud. Si vous ne savez pas le faire, voir le tuto suivant : L'utilisateur Nextcloud’s command-line

### Utiliser Nextcloud command-line

Pour avoir la liste des processus existant vous avez tout simplement besoins d'exécuter occ : `php occ`

Dans notre cas c'est l'un des scripts suivant qui nous intéresse :

```
 files
  files:cleanup                          cleanup filecache
  files:recommendations:recommend
  files:repair-tree                      Try and repair malformed filesystem tree structures
  files:scan                             rescan filesystem
  files:scan-app-data                    rescan the AppData folder
  files:transfer-ownership
```

Sur internet ils indiquent d'utiliser le script `files:scan` avec l'option `--user`, or cette option n'existe plus. Démonstration :

```
www-data@980bf1081234:~/html$ php occ files:scan --user


  The "--user" option does not exist.


files:scan [--output [OUTPUT]] [-p|--path PATH] [--all] [--unscanned] [--shallow] [--home-
```

```
only] [--] [<user_id>...]
```

Par conséquent il faudra utiliser l'une des options indiqué dans le message d'erreur. Dans notre cas nous allons utiliser l'option `[<user_id>...]`.

Pour connaître la liste des utilisateurs dans Nextcloud il faut exécuter le script occ suivant :

```
www-data@980bf1081234:~/html$ php occ user:list
  - admin: admin
  - john: john
  - guest: Invité
```

## Scanner les fichiers

Une fois que nous sommes connecté avec le bon utilisateur et que nous connaissons l'id utilisateur sur lequel nous allons exécuter le script il suffit d'effectuer la commande suivante :

```
www-data@980bf1081234:~/html$ php occ files:scan john
Starting scan for user 1 out of 1 (john)
+---------+-------+--------------+
| Folders | Files | Elapsed time |
+---------+-------+--------------+
| 13      | 30    | 00:00:01     |
+---------+-------+--------------+
```

Maintenant tous les fichiers rajouter via un transfert seront visibles sur Nextcloud.

# Rajouter une extension dans le conteneur Nextcloud

## MediaDC

source : https://github.com/andrey18106/mediadc-docker-example

Les commandes pour installer les packages nécessaires :

```
apt update
apt upgrade
apt install ffmpeg
apt install imagemagick
apt install supervisor
apt install python3-pip
```