

Nextcloud

En bref

Niveau : intermédiaire

Durée : 45 minutes

Prérequis : Traefik et Authentik en place (pages précédentes)

Niveau de test : reconstitué

Nextcloud, c'est quoi ?

Nextcloud est une plateforme de fichiers, d'agenda et de contacts auto-hébergée : l'équivalent libre d'un espace de stockage en ligne, avec synchronisation sur ordinateur et téléphone. C'est l'application phare du catalogue, et celle qui sert ici de modèle aux autres : un fichier Docker Compose, des labels Traefik, une connexion SSO créée par le rôle Authentik.

Élément	Rôle
Conteneur Nextcloud	Serveur web (Apache et PHP) : interface, API, synchronisation
Conteneur cron	Même image, exécute les tâches de fond toutes les cinq minutes
PostgreSQL	Base de données (comptes, partages, métadonnées)
Redis	Cache et verrouillage des fichiers
Application <code>user_oidc</code>	Module Nextcloud qui délègue la connexion à Authentik

Utilisateur → Traefik → Nextcloud → redirection vers Authentik → retour connecté

Concrètement, le rôle fait cinq choses dans l'ordre : déposer le fichier Compose, démarrer les conteneurs, attendre la fin de l'installation automatique, créer la connexion SSO dans Authentik, puis déclarer cette connexion dans Nextcloud avec la commande `occ`.

Cette page traite le déploiement automatisé d'une instance Nextcloud par Nébuleuse. L'installation manuelle et les commandes `occ` d'usage courant sont décrites dans le livre **Nextcloud** de la base de connaissance.

Prérequis

1. Traefik publie `cloud.penhoat.example` (variable `client_subdomains.nextcloud`) et Authentik répond sur `sso.penhoat.example`.
2. `nextcloud` figure dans `nebuleuse_apps` et dans `nebuleuse_catalogue`.
3. Deux secrets existent dans le Vault du client : `vault_nextcloud_db_password` et `vault_nextcloud_admin_password` (page « Inventaire et variables »).

Arborescence du rôle

```
roles/nextcloud/  
├─ defaults/main.yml  
├─ tasks/main.yml  
└─ templates/  
    └─ docker-compose.yml.j2
```

Variables

```
# roles/nextcloud/defaults/main.yml  
  
# Version majeure de Nextcloud : remplacer par la dernière version stable.  
# Nextcloud ne permet pas de sauter une version majeure lors d'une mise à jour.  
nextcloud_version: "32"  
  
# Dossier de travail sur la VM.  
nextcloud_dir: /opt/nebuleuse/nextcloud  
  
# Nom de domaine de l'application.  
nextcloud_host: "{{ client_subdomains.nextcloud }}.{{ client_domain }}"  
  
# Secrets : définis dans le Vault, jamais en clair ici.  
nextcloud_db_password: "{{ vault_nextcloud_db_password }}"  
nextcloud_admin_password: "{{ vault_nextcloud_admin_password }}"  
  
# Compte administrateur local, réservé à l'exploitation. Les utilisateurs du client
```

```
# se connectent par le SSO.
nextcloud_admin_user: nebuleuse-admin

# Réseaux d'où Nextcloud accepte l'en-tête X-Forwarded-For. Ici : les réseaux privés de
# Docker,
# où seul Traefik peut joindre l'application.
nextcloud_trusted_proxies: "172.16.0.0/12"

# Limites PHP : taille maximale d'un fichier envoyé et mémoire d'un processus.
nextcloud_upload_limit: 10G
nextcloud_memory_limit: 1G

# Réseau Docker partagé avec Traefik.
nextcloud_network: "{{ docker_proxy_network }}"

# Nom du fournisseur OIDC déclaré dans Nextcloud (apparaît sur le bouton de connexion).
nextcloud_oidc_provider_name: Authentik
```

Le fichier Docker Compose

```
# roles/nextcloud/templates/docker-compose.yml.j2
# Fichier géré par Ansible : toute modification manuelle sera écrasée.
# Il contient des secrets : accès réservé à root.

# Variables communes au serveur et au conteneur cron (ancrage YAML réutilisée plus bas).
x-nextcloud-env: &nextcloud_env

# Base de données.
POSTGRES_HOST: db
POSTGRES_DB: nextcloud
POSTGRES_USER: nextcloud
POSTGRES_PASSWORD: "{{ nextcloud_db_password }}"

# Cache et verrouillage.
REDIS_HOST: redis

# Lues uniquement à la première installation : créent le compte administrateur local.
NEXTCLOUD_ADMIN_USER: "{{ nextcloud_admin_user }}"
NEXTCLOUD_ADMIN_PASSWORD: "{{ nextcloud_admin_password }}"
NEXTCLOUD_TRUSTED_DOMAINS: "{{ nextcloud_host }}"

# Nextcloud est derrière Traefik : il doit construire ses adresses en HTTPS
```

```

# et faire confiance à l'en-tête X-Forwarded-For envoyé par le proxy.
OVERWRITEPROTOCOL: https
OVERWRITEHOST: "{{ nextcloud_host }}"
OVERWRITECLIURL: "https://{{ nextcloud_host }}"
TRUSTED_PROXIES: "{{ nextcloud_trusted_proxies }}"
PHP_UPLOAD_LIMIT: "{{ nextcloud_upload_limit }}"
PHP_MEMORY_LIMIT: "{{ nextcloud_memory_limit }}"

services:
  app:
    image: nextcloud:{{ nextcloud_version }}-apache
    container_name: {{ client_name }}_nextcloud
    restart: unless-stopped
    environment: *nextcloud_env
    volumes:
      # Code, configuration et données des utilisateurs.
      - {{ nextcloud_dir }}/html:/var/www/html
    depends_on:
      - db
      - redis
    networks:
      - {{ nextcloud_network }}
      - internal
    labels:
      - "traefik.enable=true"
      - "traefik.docker.network={{ nextcloud_network }}"
      - "traefik.http.routers.nextcloud-{{ client_name }}.rule=Host(`{{ nextcloud_host }}`)"
      - "traefik.http.routers.nextcloud-{{ client_name }}.entrypoints=websecure"
      - "traefik.http.services.nextcloud-{{ client_name }}.loadbalancer.server.port=80"
      # Redirige les adresses de découverte CalDAV et CardDAV vers le service DAV de
      Nextcloud.
      # "$$" : un "$" littéral dans un fichier Compose.
      - "traefik.http.routers.nextcloud-{{ client_name }}.middlewares=nextcloud-dav-{{
client_name }}"
      - "traefik.http.middlewares.nextcloud-dav-{{ client_name
}}.redirectregex.regex=https://(.*)/.well-known/(?:(card|cal)dav)"
      - "traefik.http.middlewares.nextcloud-dav-{{ client_name
}}.redirectregex.replacement=https://${1}/remote.php/dav"
      - "traefik.http.middlewares.nextcloud-dav-{{ client_name
}}.redirectregex.permanent=true"

```

cron:

```
image: nextcloud:{{ nextcloud_version }}-apache
container_name: {{ client_name }}_nextcloud_cron
restart: unless-stopped
# Script fourni par l'image : lance "cron.php" toutes les cinq minutes.
entrypoint: /cron.sh
environment: *nextcloud_env
volumes:
  - {{ nextcloud_dir }}/html:/var/www/html
depends_on:
  - db
  - redis
# Pas besoin d'être joignable : réseau interne uniquement.
networks:
  - internal
```

db:

```
image: postgres:16-alpine
container_name: {{ client_name }}_nextcloud_db
restart: unless-stopped
environment:
  POSTGRES_USER: nextcloud
  POSTGRES_DB: nextcloud
  POSTGRES_PASSWORD: "{{ nextcloud_db_password }}"
volumes:
  - {{ nextcloud_dir }}/db-data:/var/lib/postgresql/data
networks:
  - internal
```

redis:

```
image: redis:7-alpine
container_name: {{ client_name }}_nextcloud_redis
restart: unless-stopped
networks:
  - internal
```

networks:

```
{{ nextcloud_network }}:
  external: true
```

```
internal: {}
```

Seul le serveur Nextcloud est sur le réseau `proxy`. La base, Redis et le conteneur cron restent sur un réseau privé au projet. Redis n'a donc pas de mot de passe : aucun autre conteneur de la VM ne peut l'atteindre.

Déploiement

```
# roles/nextcloud/tasks/main.yml

- name: Créer les dossiers de Nextcloud
  ansible.builtin.file:
    path: "{{ item }}"
    state: directory
    mode: "0755"
  loop:
    - "{{ nextcloud_dir }}"
    - "{{ nextcloud_dir }}/db-data"
  # Le dossier "html" est créé par Docker : il appartient ensuite à l'utilisateur www-data
  # du conteneur (UID 33). Ne pas le modifier avec chown.

- name: Déposer le fichier Docker Compose de Nextcloud
  ansible.builtin.template:
    src: docker-compose.yml.j2
    dest: "{{ nextcloud_dir }}/docker-compose.yml"
    # Le fichier contient des secrets : root uniquement.
    mode: "0600"
  no_log: true

- name: Démarrer Nextcloud
  community.docker.docker_compose_v2:
    project_src: "{{ nextcloud_dir }}"
    state: present

# Au premier démarrage, l'image installe Nextcloud toute seule (plusieurs minutes).
# "occ status" ne renvoie "installed": true qu'une fois l'installation terminée.
- name: Attendre la fin de l'installation de Nextcloud
```

```

ansible.builtin.command:
  argv:
    - docker
    - exec
    - --user
    - www-data
    - "{{ client_name }}_nextcloud"
    - php
    - occ
    - status
    - --output=json
  register: nextcloud_status
# On retire les espaces pour ne pas dépendre de la mise en forme du JSON.
until: >-
  nextcloud_status.rc == 0 and
  '"installed":true' in (nextcloud_status.stdout | replace(' ', ''))
retries: 60
delay: 10
changed_when: false

# Réglages complémentaires, rejouables sans effet de bord.
- name: Régler le fonctionnement de Nextcloud
  ansible.builtin.command:
    argv: >-
      {{ ['docker', 'exec', '--user', 'www-data', client_name ~ '_nextcloud',
        'php', 'occ', 'config:system:set'] + item }}
  loop:
    # Région par défaut des numéros de téléphone.
    - ['default_phone_region', '--value=FR']
    # Fenêtre de maintenance : les tâches lourdes tournent à partir de 1 h (UTC).
    - ['maintenance_window_start', '--type=integer', '--value=1']
  changed_when: false

- name: Utiliser le conteneur cron pour les tâches de fond
  ansible.builtin.command:
    argv: [docker, exec, --user, www-data, "{{ client_name }}_nextcloud", php, occ,
background:cron]
  changed_when: false

```

Connexion SSO

Cette partie du rôle appelle le fichier de tâches de la page Authentik, puis déclare le résultat dans Nextcloud. L'adresse de retour `/apps/user_oidc/code` est celle de l'application `user_oidc`.

```
# roles/nextcloud/tasks/main.yml (suite)

- name: Créer la connexion SSO de Nextcloud dans Authentik
  ansible.builtin.include_tasks: ../../authentik/tasks/oidc_app.yml
  vars:
    oidc_app_name: Nextcloud
    oidc_app_slug: nextcloud
    oidc_redirect_uris:
      - matching_mode: strict
        url: "https://{{ nextcloud_host }}/apps/user_oidc/code"

- name: Lister les applications Nextcloud installées
  ansible.builtin.command:
    argv: [docker, exec, --user, www-data, "{{ client_name }}_nextcloud", php, occ, app:list,
--output=json]
  register: nextcloud_apps_list
  changed_when: false

- name: Installer l'application user_oidc si elle est absente
  ansible.builtin.command:
    argv: [docker, exec, --user, www-data, "{{ client_name }}_nextcloud", php, occ,
app:install, user_oidc]
  when: "'user_oidc' not in nextcloud_apps_list.stdout"

# La commande crée le fournisseur, ou le met à jour s'il existe déjà.
# --unique-uid=0      : l'identifiant Nextcloud est le nom d'utilisateur, pas un hachage.
# --mapping-uid=...  : champ du jeton utilisé comme identifiant.

- name: Déclarer Authentik comme fournisseur de connexion
  ansible.builtin.command:
    argv:
      - docker
      - exec
      - --user
```

```
- www-data
- "{{ client_name }}"_nextcloud"
- php
- occ
- user_oidc:provider
- "{{ nextcloud_oidc_provider_name }}"
- "--clientid={{ oidc_client_id }}"
- "--clientsecret={{ oidc_client_secret }}"
- "--discoveryuri={{ oidc_issuer }}.well-known/openid-configuration"
- "--unique-uid=0"
- "--mapping-uid=preferred_username"
- "--scope=openid email profile"

changed_when: false
no_log: true
```

Vérification

```
# Sur la VM : quatre conteneurs "Up".
docker ps --filter name=nextcloud --format '{{.Names}}\t{{.Status}}'

# Nextcloud est installé et en version attendue.
docker exec --user www-data alpha_nextcloud php occ status

# L'adresse publique répond et annonce une instance installée.
curl -sS https://cloud.penhoat.example/status.php | jq '{installed, maintenance, version}'

# Le fournisseur OIDC est déclaré.
docker exec --user www-data alpha_nextcloud php occ user_oidc:provider

# La redirection CalDAV fonctionne : l'adresse "Location" doit finir par /remote.php/dav.
curl -sSI https://cloud.penhoat.example/.well-known/caldav | grep -i '^location'
```

Dans le navigateur, la page `https://cloud.penhoat.example` doit proposer le bouton « Authentik ». Après connexion, l'espace fichiers du compte s'affiche. Le compte local reste accessible par `https://cloud.penhoat.example/login?direct=1`.

Pièges

1. **Variables d'installation sans effet après coup** : `NEXTCLOUD_ADMIN_*`, les paramètres de base et les domaines approuvés ne sont lus qu'à la première installation. Pour les changer ensuite, il faut passer par `occ` (par exemple `config:system:set trusted_domains 1 --value=...`).
2. « **Accès via un domaine non approuvé** » : le nom utilisé n'est pas dans `trusted_domains`. Tous les noms par lesquels l'application est atteinte doivent y figurer.
3. **Adresses en `http://` ou boucle de redirection** : `OVERWRITEPROTOCOL`, `OVERWRITEHOST` ou `TRUSTED_PROXIES` manquent. Les journaux d'accès ne montrent alors que l'adresse de Traefik.
4. **Utilisateurs créés avec un identifiant haché** : sans `--unique-uid=0`, Nextcloud nomme les comptes d'après une empreinte, illisible dans les partages et les journaux.
5. **Erreur sur l'adresse de retour** : l'adresse déclarée dans Authentik doit être exactement `https://<domaine>/apps/user_oidc/code`.
6. **Premier utilisateur SSO sans droits d'administration** : le compte du client est créé à sa première connexion, comme simple utilisateur. Pour le promouvoir : `occ group:adduser admin <identifiant>`.
7. **Tâches de fond jamais exécutées** : sans conteneur cron, le mode « AJAX » ne s'exécute que lorsqu'un utilisateur ouvre l'interface. Le conteneur cron et la commande `background:cron` règlent ce point.
8. **Gros fichiers refusés** : Traefik ne limite pas la taille, mais PHP oui. Les variables `PHP_UPLOAD_LIMIT` et `PHP_MEMORY_LIMIT` fixent les limites ; les clients de synchronisation découpent déjà les envois.
9. **Mise à jour de version** : Nextcloud n'accepte de monter que d'une version majeure à la fois. Passer de 31 à 33 impose de passer par 32.
10. **Secret client visible brièvement** : `--clientsecret` est passé en argument de `docker exec` et apparaît pendant quelques secondes dans la liste des processus de la VM, accessible à root seulement. Avec `no_log: true`, il n'apparaît pas dans les journaux d'Ansible.
11. **Nextcloud ne joint pas le SSO** : l'erreur survient à l'échange du code de connexion. Le mécanisme d'alias réseau de Traefik, décrit dans sa page, évite ce détour par l'adresse publique.

Page suivante : **Immich**.

Revision #1

Created 2026-10-17 13:54:44 UTC by clement-derouet

Updated 2026-10-17 13:54:44 UTC by clement-derouet