

Authentik (SSO)

Authentik, c'est quoi ?

Authentik est un fournisseur d'identité : il détient les comptes des utilisateurs et dit aux applications qui est connecté. Une application qui lui délègue la connexion n'a plus de mots de passe à gérer : l'utilisateur se connecte une fois sur Authentik, puis accède à toutes les applications de la VM. Le protocole utilisé est OpenID Connect (OIDC).

Élément	Rôle
Serveur Authentik	Interface web, API et pages de connexion
Worker Authentik	Tâches de fond (envoi d'e-mails, nettoyage, synchronisations)
PostgreSQL	Base de données d'Authentik (comptes, configuration)
Provider OIDC	Définit comment une application se connecte (identifiant, secret, adresses de retour)
Application	Objet Authentik qui rattache un provider à une application et à ses règles d'accès

Utilisateur → Application → redirection vers Authentik → connexion → retour à l'application avec un jeton

Concrètement, chaque application commandée par un client a besoin d'un provider et d'une application dans Authentik, créés avec les bons réglages. Faire ces réglages à la main dans l'interface est long et source d'erreurs. Le rôle les fait donc par l'API d'Authentik, avec un jeu de tâches réutilisable que toutes les pages d'application appellent.

Prérequis

1. Traefik est en place et publie `sso.alpha.nebuleuse.example` (page précédente).
2. Quatre secrets existent dans le Vault du client : `vault_authentik_secret_key`, `vault_authentik_db_password`, `vault_authentik_bootstrap_password` et `vault_authentik_bootstrap_token`. Les deux derniers complètent la liste de la page « Inventaire et variables ».
3. `curl` est installé sur la VM (fait par le socle).

Note — Le mot de passe et le jeton d'amorçage ne sont lus qu'au tout premier démarrage d'Authentik : ils créent le compte administrateur `akadmin` et son jeton d'API. Les modifier ensuite dans le Vault ne change rien dans Authentik.

Arborescence du rôle

```
roles/authentik/  
├─ defaults/main.yml  
├─ tasks/  
│   ├─ main.yml          # déploiement, attente, découverte des objets communs  
│   ├─ oidc_app.yml      # création d'un provider et d'une application OIDC (réutilisable)  
│   └─ first_user.yml    # création du premier compte du client  
└─ templates/  
    └─ docker-compose.yml.j2
```

Variables

```
# roles/authentik/defaults/main.yml  
  
# Version d'Authentik : remplacer par la dernière version stable (voir les notes de version).  
# Depuis la version 2025.10, Redis n'est plus nécessaire. Sur une version plus ancienne,  
# un conteneur Redis doit être ajouté au fichier Docker Compose.  
authentik_version: "2025.10"  
  
# Dossier de travail sur la VM.  
authentik_dir: /opt/nebuleuse/authentik  
  
# Nom de domaine d'Authentik et adresse de son API.  
authentik_host: "{{ client_subdomains.sso }}.{{ client_domain }}"  
authentik_api: "https://{{ authentik_host }}/api/v3"  
  
# Secrets : définis dans le Vault, jamais en clair ici.  
authentik_secret_key: "{{ vault_authentik_secret_key }}"  
authentik_db_password: "{{ vault_authentik_db_password }}"  
authentik_bootstrap_password: "{{ vault_authentik_bootstrap_password }}"
```

```
authentik_bootstrap_token: "{{ vault_authentik_bootstrap_token }}"

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

# Flux d'autorisation utilisé par les applications.
# "explicit-consent" : l'utilisateur valide un écran de consentement à la première connexion.
# Il existe aussi une variante "implicit-consent", sans écran de validation.
authentik_authorization_flow_slug: default-provider-authorization-explicit-consent
authentik_invalidation_flow_slug: default-provider-invalidation-flow

# Fichier d'en-tête contenant le jeton d'API (lu par curl, jamais passé en argument).
authentik_api_header_file: /root/.authentik-api-header

# Base de toutes les commandes curl qui appellent l'API.
#   -sS          : silencieux, mais affiche les erreurs
#   -k           : n'exige pas de certificat reconnu (nécessaire en mode "staging")
#   --fail-with-body : échoue sur une erreur HTTP, en affichant la réponse du serveur
#   --resolve      : résout le nom d'Authentik vers la VM elle-même, sans passer par le
DNS
#   -H @fichier    : lit l'en-tête d'authentification dans un fichier
authentik_curl_base:
  - curl
  - -sS
  - -k
  - --fail-with-body
  - --resolve
  - "{{ authentik_host }}:443:127.0.0.1"
  - -H
  - "@{{ authentik_api_header_file }}"
```

Le fichier Docker Compose

```
# roles/authentik/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 worker (ancree YAML réutilisée plus bas).
```

```
x-authentik-env: &authentik_env
  AUTHENTIK_SECRET_KEY: "{{ authentik_secret_key }}"
  AUTHENTIK_POSTGRESQL__HOST: db
  AUTHENTIK_POSTGRESQL__USER: authentik
  AUTHENTIK_POSTGRESQL__NAME: authentik
  AUTHENTIK_POSTGRESQL__PASSWORD: "{{ authentik_db_password }}"
  # Lus uniquement au premier démarrage : créent le compte "akadmin" et son jeton d'API.
  AUTHENTIK_BOOTSTRAP_PASSWORD: "{{ authentik_bootstrap_password }}"
  AUTHENTIK_BOOTSTRAP_TOKEN: "{{ authentik_bootstrap_token }}"
  AUTHENTIK_BOOTSTRAP_EMAIL: "{{ client_admin_email }}"
```

services:

server:

```
  image: ghcr.io/goauthentik/server:{{ authentik_version }}
  container_name: {{ client_name }}_authentik_server
  restart: unless-stopped
  command: server
  environment: *authentik_env
  volumes:
```

- {{ authentik_dir }}/media:/media
- {{ authentik_dir }}/custom-templates:/templates

depends_on:

db:

```
  condition: service_healthy
```

networks:

- {{ authentik_network }}
- internal

labels:

- "traefik.enable=true"
- "traefik.docker.network={{ authentik_network }}"
- "traefik.http.routers.authentik-{{ client_name }}.rule=Host(`{{ authentik\_host }}`)"
- "traefik.http.routers.authentik-{{ client_name }}.entrypoints=websecure"
- # Port HTTP interne d'Authentik.
- "traefik.http.services.authentik-{{ client_name }}.loadbalancer.server.port=9000"

worker:

```
  image: ghcr.io/goauthentik/server:{{ authentik_version }}
  container_name: {{ client_name }}_authentik_worker
  restart: unless-stopped
  command: worker
```

```

environment: *authentik_env
volumes:
  - {{ authentik_dir }}/media:/media
  - {{ authentik_dir }}/custom-templates:/templates
depends_on:
  db:
    condition: service_healthy
# Le worker n'a pas besoin d'être joignable : réseau interne uniquement.
networks:
  - internal

```

```

db:
  image: postgres:16-alpine
  container_name: {{ client_name }}_authentik_db
  restart: unless-stopped
  environment:
    POSTGRES_USER: authentik
    POSTGRES_DB: authentik
    POSTGRES_PASSWORD: "{{ authentik_db_password }}"
  volumes:
    - {{ authentik_dir }}/db-data:/var/lib/postgresql/data
# Les autres services attendent que la base soit prête avant de démarrer.
healthcheck:
  test: ["CMD-SHELL", "pg_isready -d authentik -U authentik"]
  interval: 5s
  timeout: 5s
  retries: 10
# La base n'est jamais exposée sur le réseau partagé avec Traefik.
networks:
  - internal

```

```

networks:
  {{ authentik_network }}:
    external: true
  internal: {}

```

Note — Seul le serveur est sur le réseau partagé `proxy`. La base de données et le worker restent sur un réseau privé au projet : aucune autre application de la VM ne peut atteindre la base d'Authentik.

Déploiement et découverte des objets communs

La première partie du rôle déploie les conteneurs, attend que l'API réponde, puis retrouve les objets d'Authentik dont tous les providers ont besoin : les deux flux, les mappings de scope et la règle d'accès.

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

- name: Créer les dossiers d'Authentik
  ansible.builtin.file:
    path: "{{ item }}"
    state: directory
    mode: "0755"
  loop:
    - "{{ authentik_dir }}"
    - "{{ authentik_dir }}/media"
    - "{{ authentik_dir }}/custom-templates"
    - "{{ authentik_dir }}/db-data"

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

- name: Démarrer Authentik
  community.docker.docker_compose_v2:
    project_src: "{{ authentik_dir }}"
    state: present

# Le premier démarrage applique les migrations de la base : cela peut durer plusieurs minutes.
- name: Attendre que l'API d'Authentik soit prête
  ansible.builtin.command:
    argv:
      - curl
```

```

- -fsS
- -k
- --resolve
- "{{ authentik_host }}:443:127.0.0.1"
- "https://{{ authentik_host }}/-/health/live/"
register: authentik_ready
until: authentik_ready.rc == 0
retries: 60
delay: 5
changed_when: false

# Le jeton est écrit dans un fichier d'en-tête plutôt que passé en argument à curl :
# il n'apparaît ni dans la liste des processus, ni dans les journaux d'Ansible.
- name: Déposer le fichier d'en-tête d'authentification pour l'API
  ansible.builtin.copy:
    dest: "{{ authentik_api_header_file }}"
    content: "Authorization: Bearer {{ authentik_bootstrap_token }}\n"
    owner: root
    group: root
    mode: "0600"
  no_log: true

# --- Découverte des objets communs à tous les providers ---

- name: Trouver le flux d'autorisation
  ansible.builtin.command:
    argv: "{{ authentik_curl_base + [authentik_api ~ '/flows/instances/?slug=' ~
authentik_authorization_flow_slug] }}"
  register: authentik_authz_flow_raw
  changed_when: false

- name: Trouver le flux d'invalidation
  ansible.builtin.command:
    argv: "{{ authentik_curl_base + [authentik_api ~ '/flows/instances/?slug=' ~
authentik_invalidation_flow_slug] }}"
  register: authentik_invalidation_flow_raw
  changed_when: false

- name: Lister les mappings de scope
  ansible.builtin.command:

```

```

    argv: "{{ authentik_curl_base + [authentik_api ~
'/propertymappings/provider/scope/?page_size=200'] }}"
    register: authentik_scope_mappings_raw
    changed_when: false

- name: Retenir les identifiants des objets trouvés
  ansible.builtin.set_fact:
    authentik_authz_flow_pk: "{{ (authentik_authz_flow_raw.stdout | from_json).results[0].pk
}}"
    authentik_invalidation_flow_pk: "{{ (authentik_invalidation_flow_raw.stdout |
from_json).results[0].pk }}"
    # Les trois mappings standard : sans eux, l'application reçoit un jeton sans identité.
    authentik_scope_mapping_pks: >-
    {{ (authentik_scope_mappings_raw.stdout | from_json).results
      | selectattr('scope_name', 'in', ['openid', 'profile', 'email'])
      | map(attribute='pk') | list }}

# --- Règle d'accès : "toute personne connectée peut utiliser l'application" ---

- name: Chercher la règle d'accès Nébuleuse
  ansible.builtin.command:
    argv: "{{ authentik_curl_base + [authentik_api ~ '/policies/expression/?name=nebuleuse-
allow-all'] }}"
    register: authentik_policy_raw
    changed_when: false

- name: Créer la règle d'accès si elle est absente
  ansible.builtin.command:
    argv: >-
    {{ authentik_curl_base + ['-X', 'POST', '-H', 'Content-Type: application/json',
      '--data-binary', ({'name': 'nebuleuse-allow-all', 'expression': 'return True'} |
to_json),
      authentik_api ~ '/policies/expression/'] }}"
    register: authentik_policy_created
    when: (authentik_policy_raw.stdout | from_json).results | length == 0

- name: Retenir l'identifiant de la règle d'accès
  ansible.builtin.set_fact:
    authentik_allow_policy_pk: >-
    {{ (authentik_policy_created.stdout | from_json).pk

```

```
if authentik_policy_created is not skipped
else (authentik_policy_raw.stdout | from_json).results[0].pk }}
```

Note — Les commandes `curl` sont lancées avec `argv` (une liste d'arguments) plutôt qu'avec une chaîne : il n'y a ainsi aucun guillemet à échapper dans le JSON. Dans ce déploiement, le module `ansible.builtin.uri` n'a pas été retenu : `curl` s'est montré plus prévisible, notamment avec `--resolve` et les certificats de test.

Créer une connexion OIDC pour une application

Ce fichier de tâches est le cœur de l'intégration. Il est appelé une fois par application, avec trois paramètres : le nom, l'identifiant court (le « slug ») et les adresses de retour autorisées. Il crée le provider, l'application, lie la règle d'accès, puis retourne l'identifiant client, le secret et l'adresse de l'émetteur.

```
# roles/authentik/tasks/oidc_app.yml
#
# Paramètres attendus :
#   oidc_app_name      : nom affiché dans Authentik (ex. "Nextcloud")
#   oidc_app_slug      : identifiant court, utilisé dans les adresses (ex. "nextcloud")
#   oidc_redirect_uris : liste d'adresses de retour autorisées
#                       [{ matching_mode: strict, url: "https://..." }]
#   oidc_sub_mode      : (facultatif) identifiant d'utilisateur transmis à l'application
#
# Résultats :
#   oidc_client_id, oidc_client_secret, oidc_issuer

# 1. Le provider : décrit comment l'application se connecte.
- name: "{{ oidc_app_name }}" : chercher le provider
  ansible.builtin.command:
    argv: "{{ authentik_curl_base + [authentik_api ~ '/providers/oauth2/?name=' ~
oidc_app_slug] }}"
  register: oidc_provider_lookup
  changed_when: false
  no_log: true          # la réponse contient le secret client
```

```

- name: "{{{ oidc_app_name }}} : créer le provider s'il est absent"
  ansible.builtin.command:
    argv: >-
      {{ authentik_curl_base + ['-X', 'POST', '-H', 'Content-Type: application/json',
        '--data-binary', (oidc_provider_payload | to_json),
        authentik_api ~ '/providers/oauth2/'] }}
  vars:
    oidc_provider_payload:
      name: "{{{ oidc_app_slug }}"
      # Flux d'AUTORISATION : pas le flux d'authentification, sinon l'utilisateur boucle sur
la connexion.
      authorization_flow: "{{{ authentik_authz_flow_pk }}"
      invalidation_flow: "{{{ authentik_invalidation_flow_pk }}"
      # "confidential" : l'application garde un secret côté serveur.
      client_type: confidential
      # Identifiant transmis à l'application pour désigner l'utilisateur (voir la section
Pièges).
      sub_mode: "{{{ oidc_sub_mode | default('hashed_user_id') }}"
      # Adresses de retour : "strict" exige une correspondance exacte, caractère pour
caractère.
      redirect_uris: "{{{ oidc_redirect_uris }}"
      # Les mappings openid, profile et email : sans eux, "scopes demandés non configurés".
      property_mappings: "{{{ authentik_scope_mapping_pks }}"
    register: oidc_provider_created
    when: (oidc_provider_lookup.stdout | from_json).results | length == 0
    no_log: true

- name: "{{{ oidc_app_name }}} : retenir le provider"
  ansible.builtin.set_fact:
    oidc_provider: >-
      {{ (oidc_provider_created.stdout | from_json)
        if oidc_provider_created is not skipped
        else (oidc_provider_lookup.stdout | from_json).results[0] }}
    no_log: true

# 2. L'application : rattache le provider à un nom et à des règles d'accès.
- name: "{{{ oidc_app_name }}} : chercher l'application"
  ansible.builtin.command:
    argv: "{{{ authentik_curl_base + [authentik_api ~ '/core/applications/?slug=' ~
oidc_app_slug] }}"

```

```

register: oidc_app_lookup
changed_when: false

- name: "{{ oidc_app_name }}" : créer l'application si elle est absente"
  ansible.builtin.command:
    argv: >-
      {{ authentik_curl_base + ['-X', 'POST', '-H', 'Content-Type: application/json',
        '--data-binary', (oidc_app_payload | to_json),
        authentik_api ~ '/core/applications/'] }}
  vars:
    oidc_app_payload:
      name: "{{ oidc_app_name }}"
      slug: "{{ oidc_app_slug }}"
      provider: "{{ oidc_provider.pk }}"
      policy_engine_mode: any
    register: oidc_app_created
  when: (oidc_app_lookup.stdout | from_json).results | length == 0

- name: "{{ oidc_app_name }}" : retenir l'application"
  ansible.builtin.set_fact:
    oidc_app_pk: >-
      {{ (oidc_app_created.stdout | from_json).pk
        if oidc_app_created is not skipped
        else (oidc_app_lookup.stdout | from_json).results[0].pk }}

# 3. La règle d'accès : sans elle, la connexion se termine par "Accès refusé".
# Elle n'est liée qu'à la création de l'application, pour ne pas multiplier les liens.
- name: "{{ oidc_app_name }}" : lier la règle d'accès à l'application"
  ansible.builtin.command:
    argv: >-
      {{ authentik_curl_base + ['-X', 'POST', '-H', 'Content-Type: application/json',
        '--data-binary', (oidc_binding_payload | to_json),
        authentik_api ~ '/policies/bindings/'] }}
  vars:
    oidc_binding_payload:
      target: "{{ oidc_app_pk }}"
      policy: "{{ authentik_allow_policy_pk }}"
      order: 0
  when: oidc_app_created is not skipped

```

```
# 4. Les valeurs que l'application doit connaître.
- name: "{{ oidc_app_name }}" : retenir les valeurs de connexion
  ansible.builtin.set_fact:
    oidc_client_id: "{{ oidc_provider.client_id }}"
    oidc_client_secret: "{{ oidc_provider.client_secret }}"
    # Adresse de l'émetteur : l'application y trouve la découverte automatique de la
    configuration.
    oidc_issuer: "https://{{ authentik_host }}/application/o/{{ oidc_app_slug }}/"
  no_log: true
```

Une page d'application appelle ce fichier de tâches ainsi :

```
# roles/nextcloud/tasks/main.yml (extrait)
- name: Créer la connexion SSO de Nextcloud
  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://{{ client_subdomains.nextcloud }}.{{ client_domain
        }}/apps/user_oidc/code"
```

Les adresses OIDC dont l'application a besoin sont toutes construites à partir de l'adresse de l'émetteur :

Usage	Adresse
Découverte automatique	<code>https://sso.alpha.nebuleuse.example/application/o/<slug>/well-known/openid-configuration</code>
Autorisation	<code>https://sso.alpha.nebuleuse.example/application/o/authorize/</code>
Jeton	<code>https://sso.alpha.nebuleuse.example/application/o/token/</code>
Informations sur l'utilisateur	<code>https://sso.alpha.nebuleuse.example/application/o/userinfo/</code>
Déconnexion	<code>https://sso.alpha.nebuleuse.example/application/o/<slug>/end-session/</code>

Le premier compte du client

Le compte `akadmin` sert à l'exploitation par Nébuleuse. Le client a besoin de son propre compte, créé à partir de `client_admin_email`. Son mot de passe n'est volontairement pas défini ici : il est choisi par le client, par un lien de réinitialisation envoyé par le support.

```
# roles/authentik/tasks/first_user.yml

- name: Chercher le compte du client
  ansible.builtin.command:
    argv: "{{ authentik_curl_base + [authentik_api ~ '/core/users/?username=' ~
client_admin_email] }}"
  register: authentik_user_lookup
  changed_when: false

- name: Créer le compte du client s'il est absent
  ansible.builtin.command:
    argv: >-
    {{ authentik_curl_base + ['-X', 'POST', '-H', 'Content-Type: application/json',
    '--data-binary', (authentik_user_payload | to_json),
    authentik_api ~ '/core/users/'] }}
  vars:
    authentik_user_payload:
      username: "{{ client_admin_email }}"
      name: "Administrateur {{ client_name }}"
      email: "{{ client_admin_email }}"
      is_active: true
  when: (authentik_user_lookup.stdout | from_json).results | length == 0
```

Dans `main.yml`, ce fichier est inclus à la fin du rôle :

```
# roles/authentik/tasks/main.yml (fin du fichier)

- name: Créer le premier compte du client
  ansible.builtin.include_tasks: first_user.yml
```

Vérification

```
# Sur la VM : les trois conteneurs sont démarrés et la base est « healthy ».
docker ps --filter name=authentik --format '{{.Names}}\t{{.Status}}'

# L'API répond et reconnaît le jeton : doit afficher l'utilisateur "akadmin".
```

```
curl -sSk --resolve sso.alpha.nebuleuse.example:443:127.0.0.1 \
  -H @/root/.authentik-api-header \
  https://sso.alpha.nebuleuse.example/api/v3/core/users/me/ | jq '.user.username'

# Une connexion OIDC existe et la découverte automatique répond (exemple : Nextcloud).
curl -sSk --resolve sso.alpha.nebuleuse.example:443:127.0.0.1 \
  https://sso.alpha.nebuleuse.example/application/o/nextcloud/.well-known/openid-configuration
| jq '.issuer'
```

Dans l'interface, on se connecte sur `https://sso.alpha.nebuleuse.example/if/admin/` avec `akadmin` et le mot de passe d'amorçage. Les rubriques **Applications** et **Providers** doivent lister chaque application commandée.

Pièges

1. **Flux d'authentification à la place du flux d'autorisation** : l'utilisateur se reconnecte en boucle au lieu d'être renvoyé vers l'application. Le provider doit utiliser un flux dont le slug commence par `default-provider-authorization-`.
2. **Mappings de scope absents** : l'erreur « Application requested scopes not configured » apparaît. Les trois mappings `openid`, `profile` et `email` doivent être attachés au provider.
3. « **Accès refusé** » **après une connexion réussie** : dans le déploiement qui sert de base à ce TP, une application sans règle liée refusait l'accès. La règle `return True` est liée à chaque application pour l'autoriser à toute personne connectée.
4. **sub_mode inadapté** : c'est l'identifiant de l'utilisateur transmis à l'application. La valeur par défaut `hashed_user_id` convient à la plupart des applications. Une application qui attend un UUID (PaperMerge, par exemple) échoue avec une erreur de format : utiliser `user_uuid`. Les valeurs possibles sont `hashed_user_id`, `user_id`, `user_uuid`, `user_username`, `user_email` et `user_upn`.
5. **Utilisateur introuvable côté application** : certaines applications font correspondre l'utilisateur d'Authentik à un compte qu'elles ont déjà. Le nom d'utilisateur et l'adresse e-mail doivent alors être identiques de part et d'autre, et la connexion doit se faire avec ce compte, pas avec `akadmin`.
6. **Adresse de retour différente** : l'adresse déclarée dans Authentik et celle utilisée par l'application doivent être identiques, y compris pour un `/` final. L'erreur affichée mentionne une URI de redirection invalide.
7. **Compter avec `.count`** : la liste des résultats d'Authentik est dans `.results`, et le nombre d'éléments dans `.pagination.count`. Un `.count` direct provoque l'erreur « 'dict object' has no attribute 'count' ». Le rôle compte avec `.results | length`.
8. **Jeton visible dans la liste des processus ou dans les journaux** : le passer en argument de `curl` l'expose. D'où le fichier d'en-tête en mode `0600` et les `no_log: true`.
9. **Modifier un provider existant** : le rôle ne met pas à jour un provider déjà créé. Si les adresses de retour changent, il faut les modifier dans l'interface, ou supprimer le provider

pour qu'il soit recréé.

10. **Premier démarrage long** : les migrations de la base peuvent prendre plusieurs minutes. Le rôle attend jusqu'à cinq minutes avant d'abandonner.
11. **Versión de Redis** : sur une version d'Authentik antérieure à 2025.10, le serveur et le worker exigent un conteneur Redis, absent du fichier Compose de cette page.

Pour aller plus loin

- **Remplacer le jeton d'amorçage** par un compte de service dédié, aux droits limités à la gestion des providers, puis supprimer le jeton d'amorçage une fois le déploiement terminé.
- **Configurer l'envoi d'e-mails** (SMTP) pour que les liens de réinitialisation et les notifications partent directement d'Authentik.
- **Déclarer la configuration par « blueprints »** : Authentik sait charger des fichiers de description, une alternative à l'API qui offre une vraie mise à jour déclarative.

Page suivante : **Nextcloud**.

Revision #1

Created 2026-10-16 20:34:40 UTC by clement-derouet

Updated 2026-10-16 20:38:39 UTC by clement-derouet