Déploiement d'une Infrastructure d'Auto-hébergement Souveraine

1. Objectif de la réalisation

L'objectif de ce projet est de concevoir, déployer et administrer une infrastructure informatique personnelle souveraine, divisée en deux pôles distincts :

1. **Un Serveur Multimédia & Automatisation (Pôle Cinéma) :** Déploiement d'une chaîne complète d'acquisition et de diffusion de médias (Seedbox + Media Center), hautement sécurisée par un routage VPN strict, optimisée matériellement, et totalement automatisée.
2. **Un Serveur de Services Personnels (Pôle Cloud) :** Création d'une alternative privée aux services GAFAM pour la gestion documentaire (GED) intelligente et le stockage photographique, incluant un filtrage DNS à l'échelle du réseau local.

L'enjeu principal est de garantir la confidentialité des données (anonymisation du trafic P2P), l'isolation des services via la conteneurisation, et la pérennité de l'infrastructure.

2. Moyens mis à disposition

Le projet s'articule autour de deux machines physiques distinctes et d'un environnement réseau local :

- **Serveur 1 (Pôle Cinéma) :** Machine équipée d'un processeur Intel Core i3, 24 Go de RAM.
- **Serveur 2 (Pôle Cloud) :** Machine équipée d'un processeur Intel Core i5.
- **Stockage :** Un NAS (Network Attached Storage) monté sur le réseau local pour le stockage de masse des médias (/mnt/medias_nas), complété par des disques SSD locaux pour le cache et les bases de données.
- **Environnement Système :** Linux Debian (distribution stable et légère, idéale pour l'hébergement de serveurs).

3. Solutions choisies

3.1 Applications (Front-end & Back-end métiers)

- **Pôle Cinéma :**
    - **Jellyfin :** Serveur multimédia open-source pour la diffusion de contenu.
    - **qBittorrent :** Client de téléchargement P2P.
    - **La stack "Arr" (Sonarr, Radarr, Prowlarr) :** Outils d'automatisation pour la recherche, l'acquisition et le tri des médias (Séries, Films) via l'interrogation d'indexeurs (Trackers).

- **Pôle Cloud :**
    - **Immich :** Solution de sauvegarde et d'organisation photographique avec reconnaissance faciale via Machine Learning.
    - **Paperless-ngx :** Système de Gestion Électronique de Documents (GED) avec reconnaissance optique de caractères (OCR) et IA de classification.
    - **Vaultwarden :** Gestionnaire de mot de passe compatible avec Bitwarden stocké localement
    - **AdGuard Home :** Serveur DNS menteur pour bloquer la publicité et le pistage au niveau du réseau local (LAN).

3.2 Middlewares (Intermédiaires techniques)

- **Docker & Docker Compose :** Moteur de conteneurisation choisi pour isoler chaque service, faciliter les mises à jour et unifier la gestion des dépendances.
- **Gluetun :** Client VPN léger en conteneur. Utilisé comme routeur exclusif pour le trafic P2P, faisant office de "Kill-Switch" matériel.
- **Flaresolverr :** Serveur proxy permettant de contourner les protections anti-bot (Cloudflare) des indexeurs lors des requêtes automatisées.
- **PostgreSQL & Redis :** Moteurs de base de données et gestionnaires de cache requis pour le fonctionnement d'Immich et Paperless-ngx.

3.3 Prérequis

- Configuration de baux DHCP statiques (IP fixes) sur le routeur local pour les serveurs .
- Accès SSH configuré par clé RSA pour l'administration distante sécurisée.
- Abonnement à un fournisseur VPN (ProtonVPN) supportant le protocole **WireGuard** et la fonction de **Port Forwarding** (Redirection de port).
- Montage persistant des disques réseau (NAS) via /etc/fstab sur le système hôte.

---

4. Mise en œuvre (Interface en ligne de commande - CLI)

La mise en œuvre a débuté par la préparation de l'hôte Debian et l'installation du moteur Docker.

Étape 1 : Préparation de l'environnement

Bash

```
# Mise à jour des paquets et installation des prérequis

sudo apt update && sudo apt upgrade -y

sudo apt install -y curl git ufw cifs-utils
```

Installation de Docker Engine via le script officiel

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

Création de l'arborescence des dossiers de configuration

mkdir -p /opt/cinema/downloaders/qbit_config

mkdir -p /mnt/services_docker/cinema/data/torrents

Étape 2 : Configuration du réseau Docker

Création d'un réseau bridge dédié pour permettre aux conteneurs de communiquer entre eux :

Bash

docker network create cinema_net

Étape 3 : Déploiement de la stack sécurisée (Extrait docker-compose.yml)

La complexité résidait dans le couplage de qBittorrent au réseau de Gluetun pour garantir l'anonymat (Kill-Switch). *Exemple de configuration (clés censurées) :*

YAML

```
services:
  gluetun:
        image: qmcgaw/gluetun
        container_name: gluetun
        cap_add:
        - NET_ADMIN
        environment:
        - VPN_SERVICE_PROVIDER=protonvpn
```

```
      - VPN_TYPE=wireguard
      - VPN_PORT_FORWARDING=on
      - WIREGUARD_PRIVATE_KEY=[Ma clef privée]
      - SERVER_COUNTRIES=France
      - TZ=Europe/Paris
      ports:
      - 8080:8080 # Interface qBittorrent mappée sur le VPN
      - 6881:6881 # Port P2P TCP/UDP
      - 6881:6881/udp
      restart: unless-stopped
      networks:
      - cinema_net

qbittorrent:
      image: lscr.io/linuxserver/qbittorrent:latest
      container_name: qbittorrent
      environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Paris
      - WEBUI_PORT=8080
      volumes:
      - /opt/cinema/downloaders/qbit_config:/config
      - /mnt/services_docker/cinema/data/torrents:/data/torrents
      # ROUTAGE STRICT : Asservissement au réseau de Gluetun
      network_mode: "service:gluetun"
      depends_on:
```

```
        gluetun:
        condition: service_healthy
        restart: unless-stopped
```

Étape 4 : Lancement et vérification

Bash

```
# Déploiement en tâche de fond
docker compose up -d

# Vérification de quel port est donné par le vpn
docker logs gluetun | grep -i "port forwarded"
# Résultat attendu : "port forwarded is XXXXX"
```

Le lancement du service de téléchargement est le plus complexe, le reste se fait plutot simplement, il faut juste penser dans mon cas a rajouter le bloc:

```
tmpfs:
    - /transcode
```

dans le docker de jellyfin pour pouvoir activer le transcodage video via la RAM plus tard

Étape 5 : Libération du port DNS (Serveur i5 - AdGuard) Pour le serveur i5, le service systemd-resolved de Debian bloquait le port 53. Il a fallu le désactiver pour installer AdGuard :

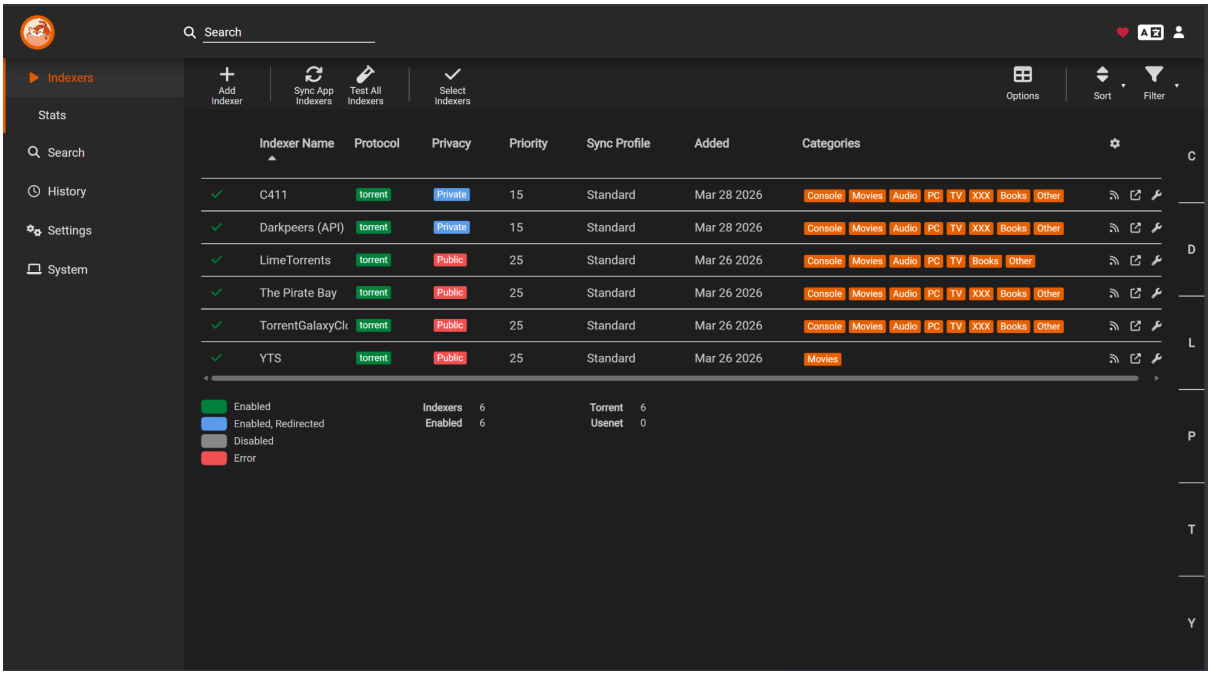
Bash

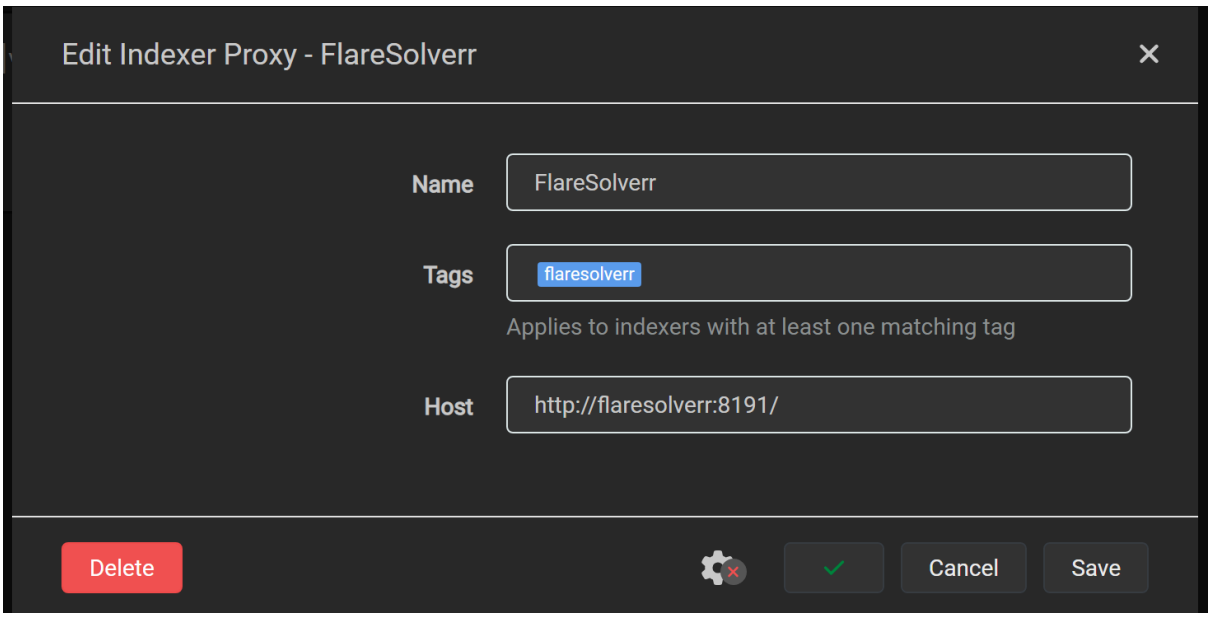
```
sudo systemctl disable systemd-resolved

sudo systemctl stop systemd-resolved

sudo ss -tulpn | grep :53 # Vérification de la libératio
```

5. Mise en œuvre côté interface WEB (Configuration Applicative)

5.1 Pôle Cinéma (La Stack d'Automatisation)

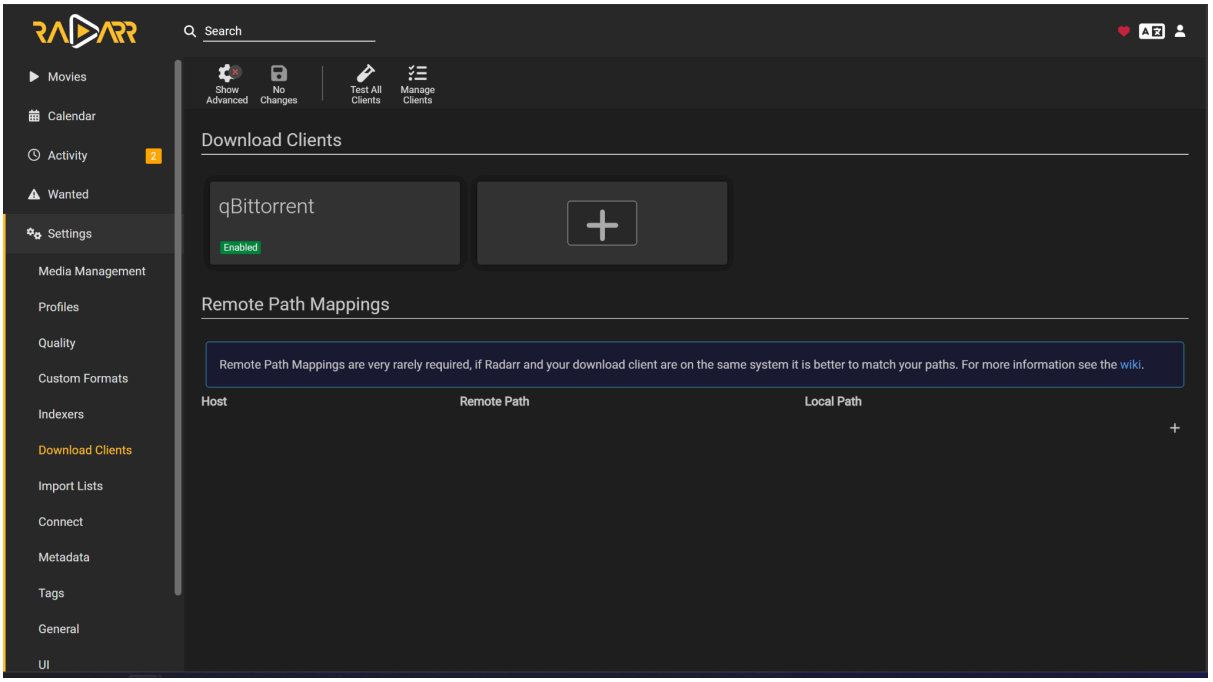
1. Prowlarr (Le chef d'orchestre des indexeurs)
    - Contournement anti-bot : Interfaçage avec le proxy Flaresolverr. Ce dernier a été assigné via un système de "Tags" aux indexeurs francophones stricts pour résoudre les défis Cloudflare.

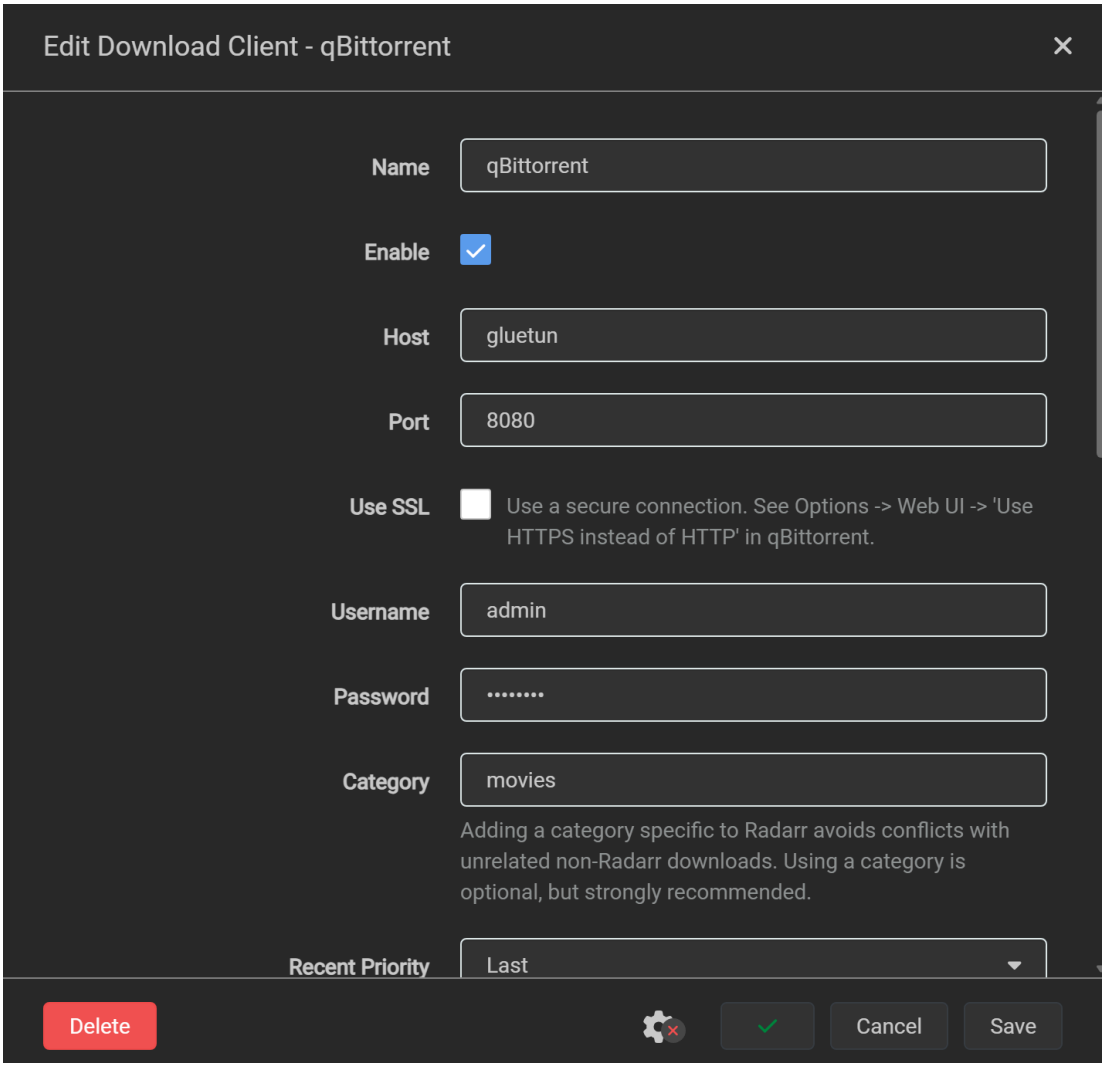




- Synchronisation API : Ajout des clés API de Sonarr et Radarr pour l'injection automatique des indexeurs.

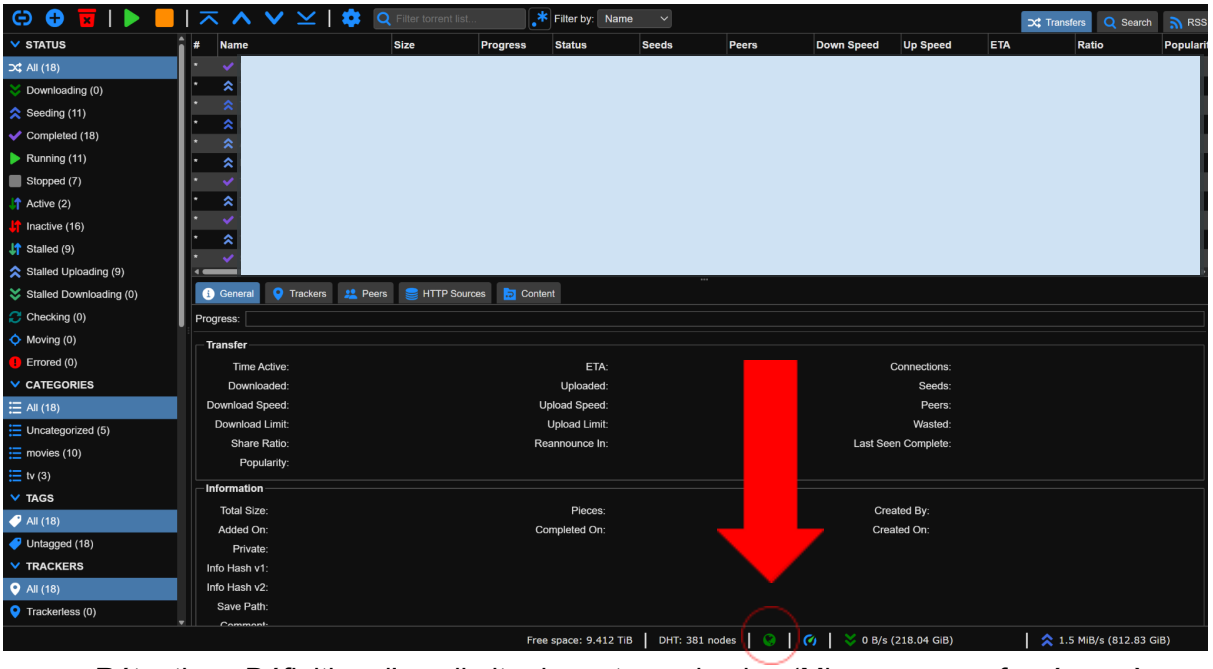
2. Radarr (Sonarr se configure exactement de la même manière)
   - Root Folders : Mapping des chemins de destination vers le NAS (/data/torrents/movies).
   - Et le download client a configurer





3. Bazarr (Gestion des sous-titres)
    - Interfaçage : Connexion aux API de Sonarr et Radarr pour rechercher les sous-titres FR/EN manquants avec un score de synchronisation strict, configuration très simple, une fois le reste de la stack faite.

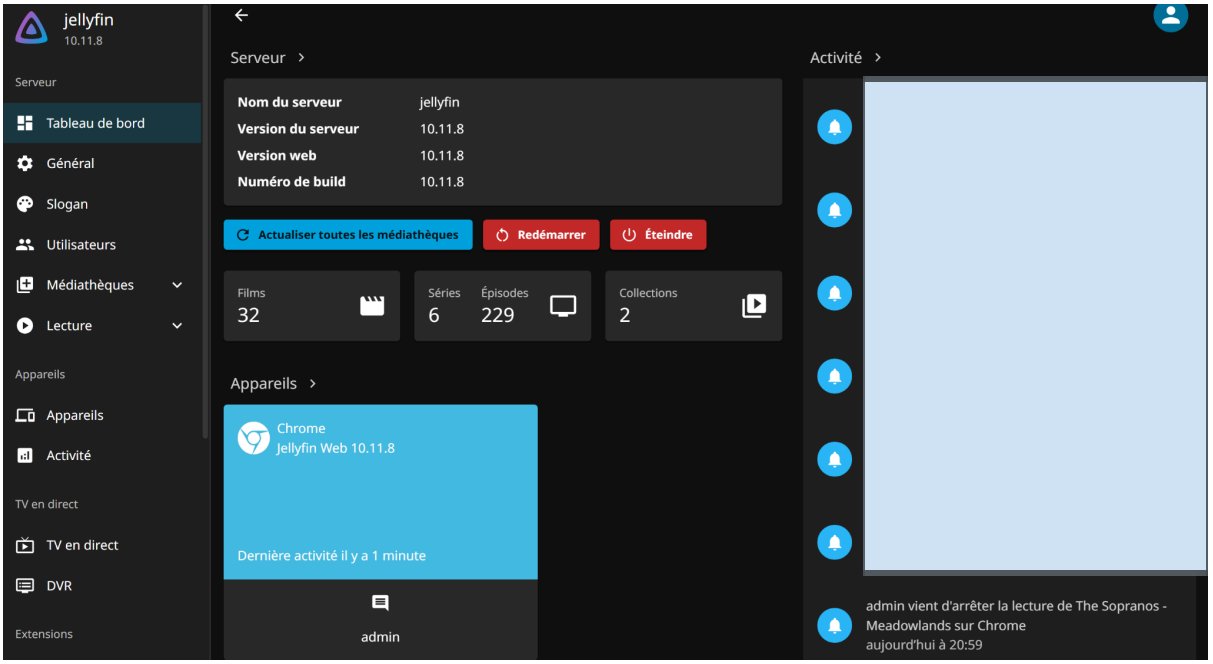
4. qBittorrent (Le client de téléchargement)
    - Optimisation du Seeding : Intégration manuelle du port dynamique généré par Gluetun pour passer en peering "Actif".



- Rétention : Définition d'une limite de partage absolue (Mise en pause forcée après 10080 minutes / 7 jours de seed).

5. Jellyfin & Jellyseerr (Le Front-end utilisateur)

- Jellyfin - Transcodage Matériel : Configuration du dossier de transcodage vers /dev/shm (RAM) pour préserver le SSD.
- Jellyfin - Génériques : Abaissement du Pourcentage de lecture maximum pour la reprise à 80% pour éviter la pollution de l'accueil.

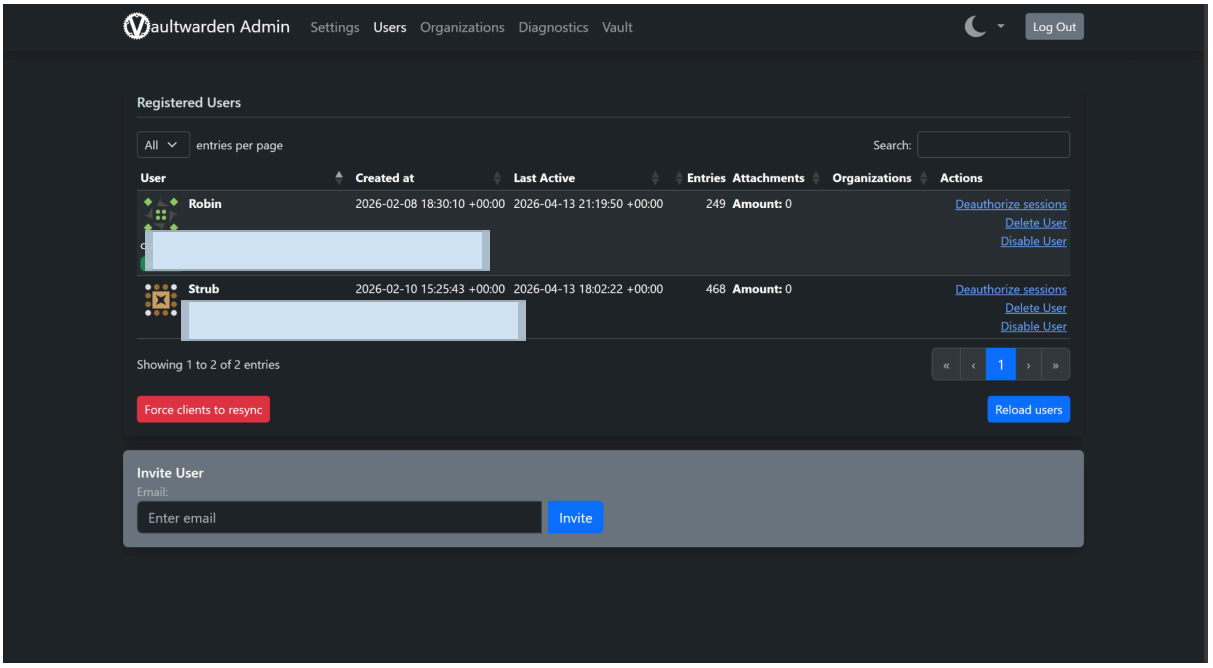


- Jellyseerr : Configuration du SSO (connexion avec identifiants Jellyfin) et routage des requêtes vers Radarr/Sonarr.

5.2 Pôle Cloud (Services Personnels)

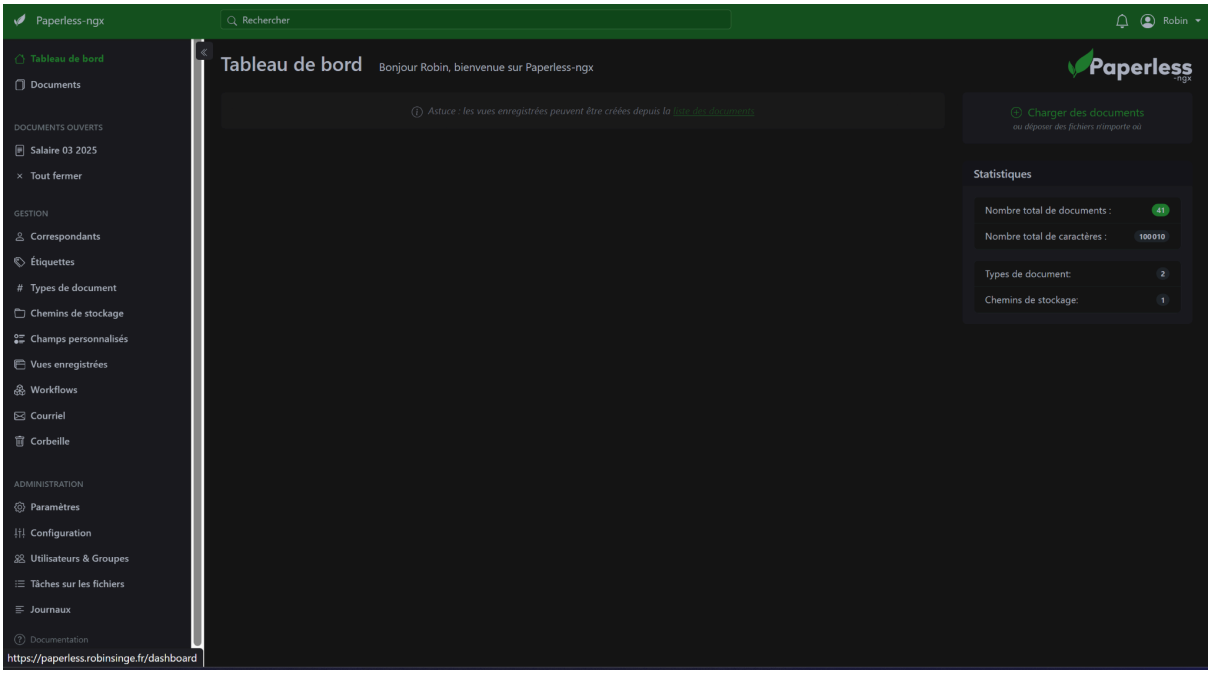
1. Vaultwarden (Gestionnaire de mots de passe)

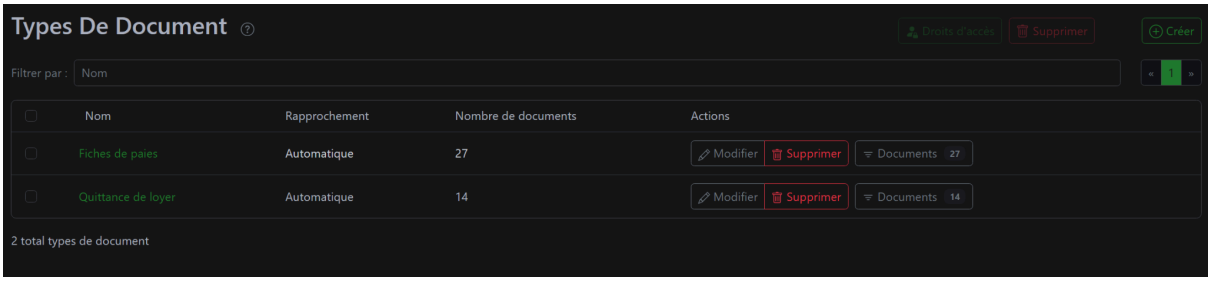
- Déploiement du conteneur. Accès au panneau d'administration via un ADMIN_TOKEN robuste généré par commande cryptographique.



2. Immich (Sauvegarde photographique IA)
   - Paramétrage du Storage Template pour organiser les fichiers physiques sur le disque (Année/Mois/Jour).
   - Configuration matérielle (QuickSync sur le processeur i5) pour l'accélération du Machine Learning (Reconnaissance faciale et d'objets).

3. Paperless-ngx (GED)
   - Paramétrage du moteur Tesseract pour la reconnaissance OCR franco-anglaise.
   - Mise en place de règles de correspondance (Auto/Regex) pour le taggage automatique.





L’OCR ayant besoin de beaucoup de données, pour automatiser le tri , je n’ai que deux pièces d’identités scannées celles ci sont donc triés manuellement , mais pour les documents que j’ai en dizaines d’exemplaires le tri se fait parfaitement

L'administration et l'optimisation des services se sont faites via leurs interfaces web respectives (GUI).

- **Optimisation du Seeding (qBittorrent) :** Récupération du port dynamique généré par Gluetun en CLI, et intégration manuelle dans l'interface de qBittorrent (*Outils > Connexion > Port utilisé pour les connexions entrantes*). Cela permet de passer d'un état "Passif" à "Actif", maximisant ainsi la connectivité P2P.

- **Cycle de vie des données (Rétention) :** Configuration d'une limite de partage dans qBittorrent (Mise en pause forcée après 10080 minutes / 7 jours de seed). Côté Sonarr/Radarr, activation de l'option *Remove Completed* pour détruire automatiquement le fichier source une fois le quota atteint, évitant la saturation des disques.

- **Configuration des Indexeurs (Prowlarr / Sonarr) :** Ajout du tag flaresolverr sur les indexeurs stricts (ex: C411) pour déléguer la résolution des captchas Cloudflare. Configuration spécifique des catégories ("Anime" = 5070) pour garantir le matching des recherches automatisées.

- **Optimisation de l'Expérience Multimédia (Jellyfin) :**
    - **Gestion des génériques :** Abaissement du paramètre *Pourcentage de lecture maximum pour la reprise* à 80% pour éviter que la section "Continuer à regarder" ne soit polluée par des épisodes dont seul le générique de fin reste à visionner.

    - **Transcodage RAM :** Configuration du dossier de transcodage matériel vers /dev/shm (mémoire vive) pour utiliser les 24 Go de RAM et préserver le SSD de l'usure des cycles d'écriture.

---

6. Problèmes rencontrés et réussites

Problèmes techniques et résolutions (Troubleshooting)

1. **Instabilité du conteneur de synchronisation de port (Docker Hub) :**
    - *Symptôme :* Échec de la commande docker compose up -d avec l'erreur pull access denied lors de la tentative d'automatisation de la synchronisation entre Gluetun et qBittorrent (images spx01 et casvt).
    - *Cause :* Purge des dépôts inactifs par Docker Hub, rendant les images communautaires indisponibles.
    - *Résolution :* Retour à une configuration manuelle stable. Le port VPN ProtonVPN étant statique sur de longues périodes, la mise à jour se fait manuellement via les logs Gluetun en cas de redémarrage complet du serveur.
2. **Échec des recherches Sonarr sur certains Trackers (C411) :**
    - *Symptôme :* Les requêtes de téléchargement d'animés échouaient ou mettaient l'indexeur en quarantaine (Backoff penalty).
    - *Cause :* Protection Cloudflare + requêtes orientées par défaut sur la catégorie "Séries TV" (5000) au lieu de la catégorie spécifique au site.
    - *Résolution :* Routage des requêtes via Flaresolverr et forçage manuel de la catégorie 5070 (Anime) dans Prowlarr. Mise en place d'une limite de requêtes (100/Jour) pour éviter les bannissements.
3. **Bugs d'interface sur le client Jellyfin (Recherche infinie et erreurs de langue) :**
    - *Symptôme 1 :* Erreur Invalid language tag: null sur l'application mobile.
    - *Résolution 1 :* Désactivation de la détection "Auto" et forçage du profil utilisateur en "Français" côté serveur pour contourner les protections anti-pistage des navigateurs mobiles.
    - *Symptôme 2 :* Le moteur de recherche tournait à l'infini (timeout).
    - *Résolution 2 :* Identification d'une mise en veille prolongée des disques du NAS (spin-down) et/ou verrouillage SQLite. Résolu par un docker restart et une reconstruction forcée de l'index des métadonnées.

Réussites

- Mise en place d'un **Kill-Switch réseau infaillible** (via le network_mode: service:gluetun), rendant toute fuite d'IP impossible.
- Déploiement réussi d'une architecture souveraine de GED (Paperless) et de sauvegarde photo (Immich), éliminant la dépendance aux services cloud tiers et très agréablement surpris par l'utilité de Paperless pour les documents administratifs, simplicité d'usage remarquable ainsi qu'une utilité ressentie dès la premiere semaine avec le léger modèle d'IA qui tourne avec l'OCR pour trier automatiquement les documents (factures triées par debiteur, fiches de paies, pièces d'identités, certificats).
- Optimisation matérielle poussée (Transcodage en RAM, Port Forwarding dynamique) garantissant des performances équivalentes à une Seedbox commerciale professionnelle et qui permet une lecture de qualité et très rapide sur le jellyfin.

---

7. Conclusion et Remerciements

Ce projet d'infrastructure système m'a permis de mettre en œuvre des concepts avancés d'administration réseau et de conteneurisation. En partant de machines physiques vierges, j'ai pu déployer deux environnements distincts, stables et sécurisés.

L'utilisation de Docker et Docker Compose a prouvé son efficacité pour la modularité du système, permettant d'isoler des flux réseau complexes (comme le tunnel WireGuard) tout en facilitant la maintenance. Les défis rencontrés, notamment autour des limites des images communautaires (Docker Hub) ou des subtilités du Port Forwarding dynamique, ont exigé un processus rigoureux de diagnostic (lecture des logs, tests réseau) et ont grandement renforcé mes compétences en résolution d'incidents (troubleshooting).

Aujourd'hui, l'infrastructure répond à 100 % au cahier des charges : elle est hautement disponible, respecte la confidentialité des flux de données, et gère le cycle de vie du stockage de manière totalement autonome, je l’utilise quotidiennement et ait fourni des accès a mes proches qui sont aussi satisfait des services proposés.

Remerciements:
Je tiens à remercier mon camarade Julien Ourliac pour ce projet, il avait déja un homelab en service et m’as grandement inspiré pour réaliser le mien, de plus il m’as beaucoup aidé non seulement en me fournissant des conseils techniques mais aussi en me vendant un NAS a très bon prix qui était la seule pièce manquante. J’aimerais aussi remercier mon père qui me laisse laisser mon infrastructure chez lui, mon studio n’étant pas propice a héberger ce genre de machines et enfin mon ami Marius Capelli très doué en développement qui m’as aidé pour des scripts d’automatisations sur mes machines et qui me pousse à progresser en informatique en général depuis le lycée.

P.S: Tous les médias téléchargés sur le jellyfin sont soit libres de droits, soit je possède le Blu-Ray.