Docker

Docker est une plateforme permettant de lancer certaines applications dans des conteneurs

- Règles IPTables pour Docker
- Compose
  - PaperMC Minecraft Serveur
  - Ollama avec Open WebUI
  - h5ai
  - Syncthing
  - Bookstack
  - librespeed
  - Password Pusher
  - Restreamer
  - Jellyfin avec transcodage matériel NVIDIA
  - JellyStat
  - Jellyseerr
  - Joal
  - Stable Diffusion web UI NVIDIA
  - Exemple de documentation : NomDuService
  - PhantomBot
  - yt-dlp Web UI
  - handbrake
  - Téléchargez avec qBittorrent et boostez votre ratio grâce à Joal, en restant derrière un VPN
  - Watchtower
  - Zabbix
  - Effacement sécurisé complet du disque

Règles IPTables pour Docker

Par défaut, Docker autorise toutes les adresses IP externes à se connecter aux conteneurs, pour restreindre l'accès à une seule IP ou à un réseau, il faut ajouter une règle dans la chaîne `DOCKER-USER`, située avant celles gérées par Docker dans la chaîne `DOCKER`, sans modifier directement ces dernières. Notez que les exemples concernent l'IPv4 : pour une infrastructure en IPv6, il convient d'appliquer des règles similaires avec `ip6tables`, afin de couvrir à la fois IPv4 (`iptables`) et IPv6 (`ip6tables`).

Étapes pour appliquer les règles après le démarrage de Docker

Créez un fichier script, par exemple `/usr/local/bin/iptables-rules.sh` :

```
sudo nano /usr/local/bin/iptables-rules.sh
```

Ajouter les règles dans le script :

```
#!/bin/bash
# Remise à zéro des règles personnalisées Docker
iptables -F DOCKER-USER
ip6tables -F DOCKER-USER

# Autorise l'accès au port 9443 uniquement depuis 111.11.111.111
iptables -A DOCKER-USER -p tcp --dport 9443 -s 111.11.111.111 -j ACCEPT

# Autorise l'accès public aux ports 80 (HTTP) et 443 (HTTPS)
iptables -A DOCKER-USER -p tcp --dport 80 -j ACCEPT
iptables -A DOCKER-USER -p tcp --dport 443 -j ACCEPT

# Autorise tout le trafic sortant des containers (nécessaire pour apt/pip/npm, etc.)
iptables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A DOCKER-USER -o ens16 -j ACCEPT

# Bloque tout le reste (en dernier)
iptables -A DOCKER-USER -j DROP
```

```
# ================== Pour IPv6 ==================

# Autorise l'accès public aux ports 80 et 443 en IPv6
ip6tables -A DOCKER-USER -p tcp --dport 80 -j ACCEPT
ip6tables -A DOCKER-USER -p tcp --dport 443 -j ACCEPT

# Autorise le trafic sortant IPv6
ip6tables -A DOCKER-USER -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
ip6tables -A DOCKER-USER -o ens16 -j ACCEPT

# Bloque tout le reste
ip6tables -A DOCKER-USER -j DROP
```

Rendre le script exécutable :

```
sudo chmod +x /usr/local/bin/iptables-rules.sh
```

Ci-dessous une version qui lit un fichier de whitelist peut être parfaite pour du proxy Cloudflare :

```
#!/bin/bash

# Remise à zéro des règles personnalisées Docker
iptables -F DOCKER-USER
ip6tables -F DOCKER-USER

# Remplacer 'ext_if' par le nom de votre interface réseau externe, comme 'eth0'.
ext_if="eth0"  # Remplacez par votre interface réseau externe
whitelist_file="/usr/local/bin/whitelist.txt"  # Chemin vers le fichier de liste blanche

# Vérifier que le fichier de liste blanche existe et est accessible en lecture
if [ ! -r "$whitelist_file" ]; then
    echo "Erreur : Le fichier de liste blanche '$whitelist_file' n'existe pas ou n'est pas
accessible en lecture." >&2
    exit 1
fi

# Lire le fichier de liste blanche et ajouter des règles pour chaque adresse IP ou plage CIDR
while read -r line; do
    # Ignorer les lignes de commentaire et les lignes vides
    if [[ ! "$line" =~ ^# ]] && [[ -n "$line" ]]; then
```

```
        # Détecter si l'adresse est IPv4 ou IPv6
        if grep -qE ':' <<< "$line"; then
            # Adresse IPv6
            ip6tables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p tcp -m multiport --dports
80,443 -j DROP
            ip6tables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p udp -m multiport --dports
80,443 -j DROP
        else
            # Adresse IPv4
            iptables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p tcp -m multiport --dports
80,443 -j DROP
            iptables -I DOCKER-USER -i "$ext_if" ! -s "$line" -p udp -m multiport --dports
80,443 -j DROP
        fi
    fi
done < "$whitelist_file"
```

Crée la whitlist (pour l'exemple du proxy Cloudflare) :

```
sudo nano /usr/local/bin/whitelist.txt
```

```
# Cloudflare https://www.cloudflare.com/fr-fr/ips/
173.245.48.0/20
103.21.244.0/22
103.22.200.0/22
103.31.4.0/22
141.101.64.0/18
108.162.192.0/18
190.93.240.0/20
188.114.96.0/20
197.234.240.0/22
198.41.128.0/17
162.158.0.0/15
104.16.0.0/13
104.24.0.0/14
172.64.0.0/13
131.0.72.0/22
2400:cb00::/32
2606:4700::/32
2803:f800::/32
```

```
2405:b500::/32
2405:8100::/32
2a06:98c0::/29
2c0f:f248::/32
```

Créer un service systemd pour exécuter ce script avant Docker :

```
sudo nano /etc/systemd/system/iptables-rules.service
```

Contenu du fichier :

```
[Unit]
Description=Apply custom iptables rules
After=docker.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/iptables-rules.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

Activer le service pour qu'il s'exécute au démarrage avant Docker :

```
sudo systemctl daemon-reload
sudo systemctl enable iptables-rules.service
```

(Re)démarrez le service système du pare-feu et Docker :

```
sudo systemctl restart iptables-rules.service
sudo systemctl restart docker
```

Pour tester la bonne application des règles, redémarrez le serveur puis vérifiez les logs des services pour vous assurer que tout fonctionne correctement :

```
sudo reboot
```

Après le redémarrage, consultez les logs du service IPTables personnalisé et du service Docker :

```
sudo journalctl -u iptables-rules.service
sudo journalctl -u docker.service
```

Documentation

- https://www.n0tes.fr/2019/05/11/Docker-et-IPtables/
- https://docs.docker.com/network/iptables/#restrict-connections-to-the-docker-daemon
- https://docs.docker.com/v17.09/engine/userguide/networking/default_network/custom-docker0/

Compose

Compose

PaperMC Minecraft Serveur

Serveur Minecraft

```
# Crée un volume Docker nommé "minecraftserver"
docker volume create minecraftserver
```

```
services:
  minecraft:
    # Utilise l'image "marctv/minecraft-papermc-server" avec la version 1.21.4 de Minecraft
    image: marctv/minecraft-papermc-server:1.21.4
    # Redémarre le conteneur en cas de défaillance
    restart: always
    # Définit le nom du conteneur
    container_name: "mcserver"
    environment:
      # Définit la RAM maximum pour le conteneur
      MEMORYSIZE: "2G"
      PAPERMC_FLAGS: ""
    volumes:
      - minecraftserver:/data
    # Ouvre le port 25565 du conteneur et le redirige vers le port 25565 de l'hôte
    ports:
      - "25565:25565"
    networks:
      - minecraft_network
    stdin_open: true
    tty: true
networks:
  minecraft_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
volumes:
  minecraftserver:
    external: true
```

Pour accéder à la console shell et modifier les fichiers du serveur Minecraft :

```
# Exécute la commande "/bin/bash" dans le conteneur "mcserver"
docker exec -it mcserver /bin/bash
```

Installer un éditeur de texte

```
apt update && apt install nano
```

Pour copier un fichier du serveur Minecraft dans un autre conteneur Docker. Le dossier des fichiers se trouve à /data :

```
docker cp <source> mcserver:/data/<destination>
```

Pour copier un répertoire du serveur Minecraft dans un autre conteneur Docker de manière récursive. Le dossier des fichiers se trouve à /data :

```
docker cp -r <source> mcserver:/data/<destination>
```

Compose

Ollama avec Open WebUI

1. Installation de la boîte à outils NVIDIA Container
2. Crée un volume pour le conteneur :

```
docker volume create ollama
docker volume create open-webui
```

```
services:
  ollama:
    container_name: ollama
    image: ollama/ollama:latest
    restart: 'unless-stopped'
    ports:
      - '11434:11434'
    volumes:
      - 'ollama:/root/.ollama'
    environment:
      TZ: 'Europe/Zurich'
      PUID: '1000'
      PGID: '1000'
      OLLAMA_HOST: '0.0.0.0'
      NVIDIA_DRIVER_CAPABILITIES: 'all'
      NVIDIA_VISIBLE_DEVICES: 'all'
    deploy:
      resources:
        limits:
          cpus: '2.00'
          memory: '4096M'
        reservations:
          cpus: '1.00'
          memory: '4096M'
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
```

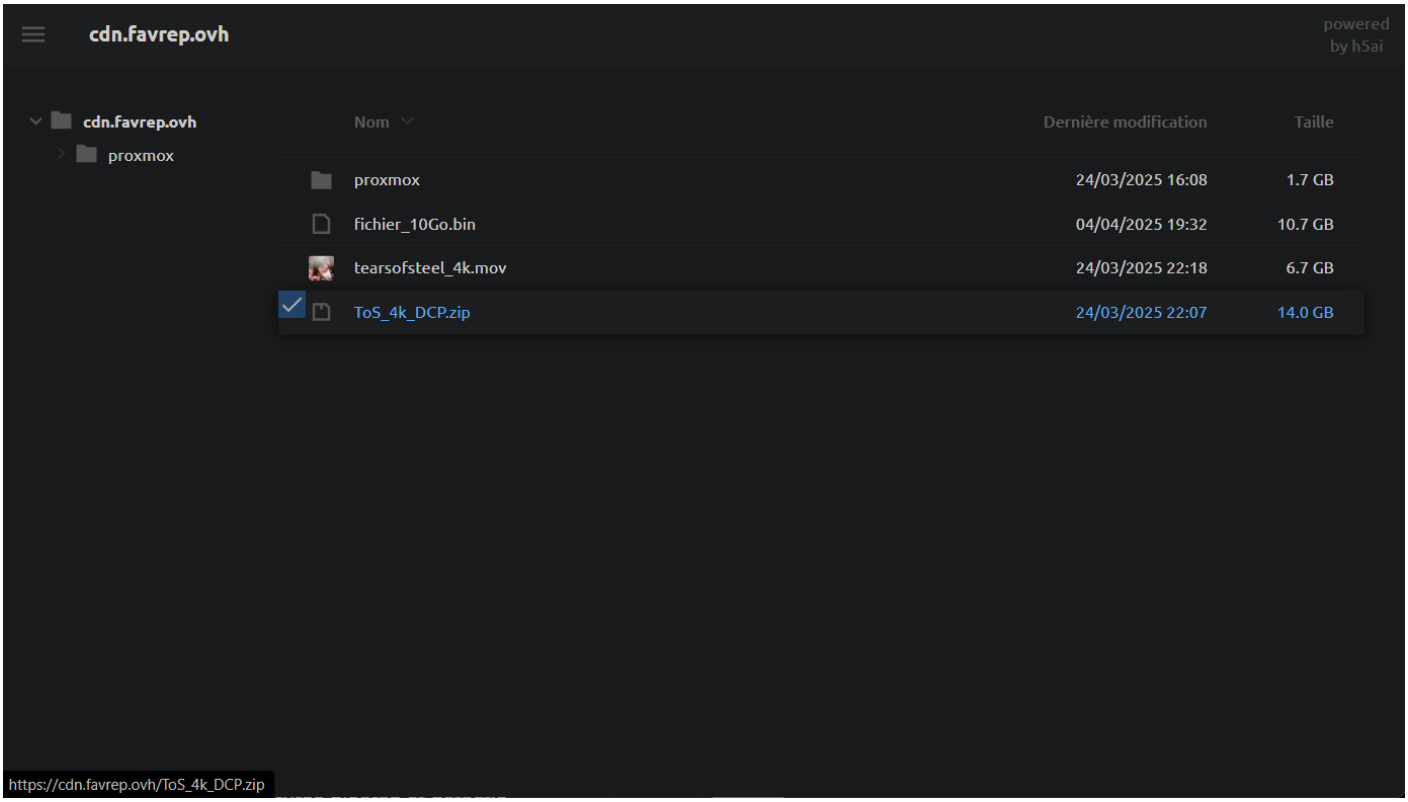
```
  open-webui:
    container_name: open-webui
    image: ghcr.io/open-webui/open-webui:${WEBUI_DOCKER_TAG-main}
    restart: 'unless-stopped'
    volumes:
      - 'open-webui:/app/backend/data'
volumes:
  ollama:
  open-webui:
    external: true
```

Pour voir l'utilisation en temps réelle sur l'hôte :

```
nvidia-smi dmon
```

Compose

h5ai




```
docker volume create h5ai_config
docker volume create h5ai_shared
```

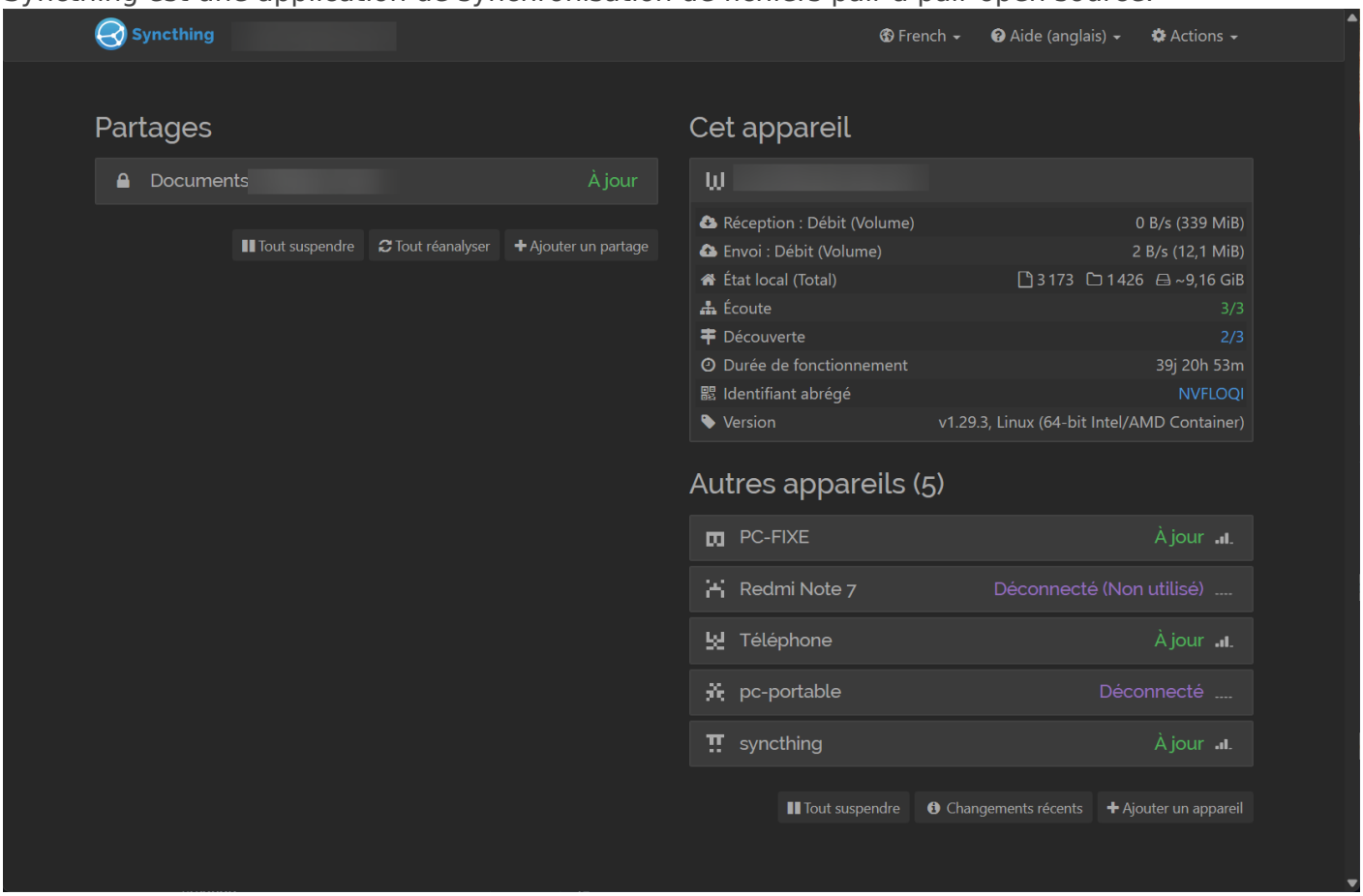
```
services:
  h5ai:
    image: awesometic/h5ai
    container_name: cdn_h5ai
    environment:
      - TZ=Europe/Zurich
      - PGID=$GID
      - PUID=$UID
    volumes:
      - h5ai_config:/config
      - h5ai_shared:/h5ai
    ports:
      - '82:80'
    restart: always
```

```
    tty: true
    stdin_open: true
networks:
  h5ai_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
volumes:
  h5ai_config:
    external: true
  h5ai_shared:
    external: true
```

Compose

Syncthing

Syncthing est une application de synchronisation de fichiers pair à pair open source.




```
docker volume create syncthing_config
docker volume create syncthing_data1
```

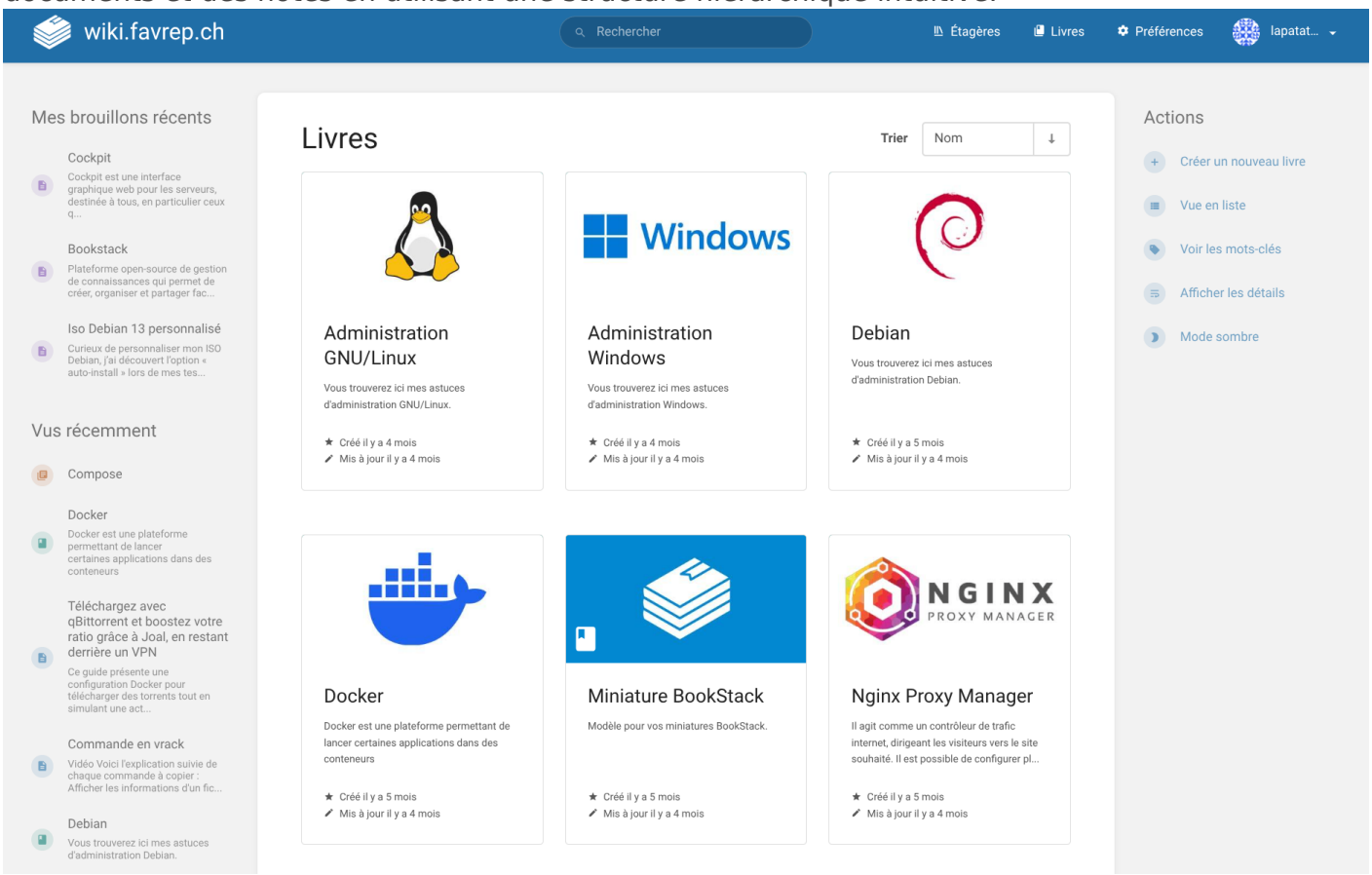
```
services:
  syncthing:
    image: lscr.io/linuxserver/syncthing:latest
    container_name: syncthing
    hostname: syncthing.favrep.ovh
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Zurich
    volumes:
      - syncthing_config:/config
```

```
      - syncthing_data1:/data1
    ports:
      - 8384:8384               # Application WebUI
      - 22000:22000/tcp         # Listening port (TCP)
      - 22000:22000/udp         # Listening port (UDP)
      - 21027:21027/udp         # Protocol discovery
    restart: unless-stopped
networks:
  syncthingn_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
volumes:
  syncthing_config:
      external: true
  syncthing_data1:
      external: true
```

Compose

Bookstack

Plateforme de gestion de connaissances qui permet de créer, organiser et partager facilement des documents et des notes en utilisant une structure hiérarchique intuitive.

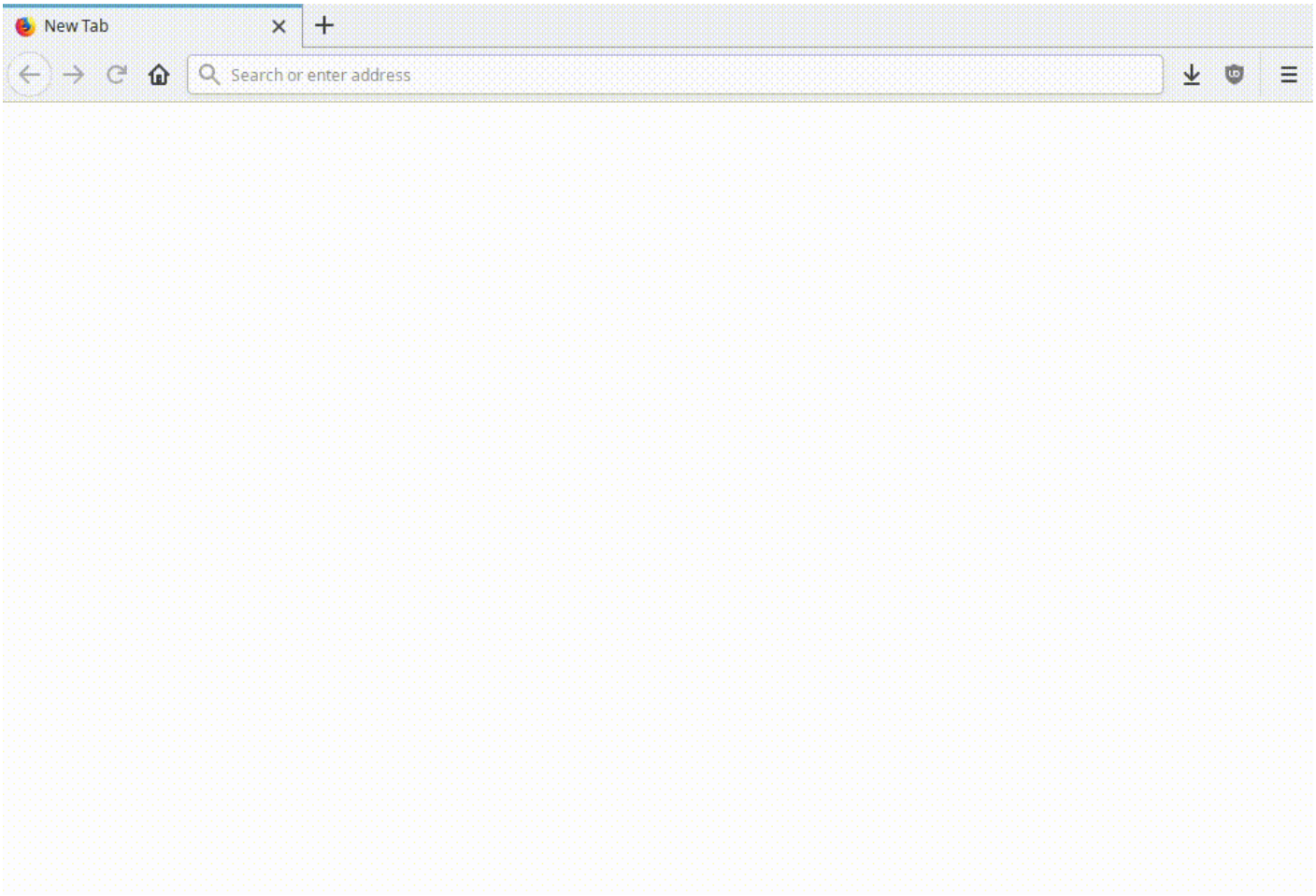



```
services:
  bookstack:
    image: lscr.io/linuxserver/bookstack:latest
    container_name: bookstack
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
      - APP_KEY=base64:mhoEccThD5nT/94EWcFnWQV6E8XNeNVca75J7neYz6s=
      - APP_URL=http://192.168.1.118:6875      # Vérifiez que le dossier correspond à votre
configuration
      - DB_HOST=bookstack_db
      - DB_PORT=3306
      - DB_USERNAME=bookstack
```

```
      - DB_PASSWORD=VotreMotDePasse           # Remplacez le mot de passe
      - DB_DATABASE=bookstackapp
      - APP_DEFAULT_DARK_MODE=true
      #- MAIL_DRIVER=smtp
      #- MAIL_HOST=smtp.gmail.com
      #- MAIL_PORT=465
      #- MAIL_ENCRYPTION=tls
      #- MAIL_USERNAME=yourEMAIL
      #- MAIL_PASSWORD=yourPassword
      #- MAIL_FROM=yourEMAIL
      #- MAIL_FROM_NAME=yourNAME
    volumes:
      - /bookstack/config:/config             # Vérifiez que le dossier correspond à votre
configuration
    ports:
      - 6875:80
    restart: unless-stopped
  bookstack_db:
    image: lscr.io/linuxserver/mariadb:latest
    container_name: maria_db
    ports:
      - 3308:3306                             # 3308 is a DB port visible on HOST
    environment:
      - PUID=1000
      - PGID=1000
      - MYSQL_ROOT_PASSWORD=VotreMotDePasseBis # Remplacez le mot de passe
      - MYSQL_DATABASE=bookstackapp
      - MYSQL_USER=bookstack
      - MYSQL_PASSWORD=VotreMotDePasse         # Remplacez le mot de passe
    volumes:
      - /bookstack_db/config:/config          # Vérifiez que le dossier correspond à votre
configuration
    restart: unless-stopped

# Connexion par défaut :
# Email : admin@admin.com
# Mot de passe : password
```

Compose

librespeed




```
docker volume create librespeed_config
```

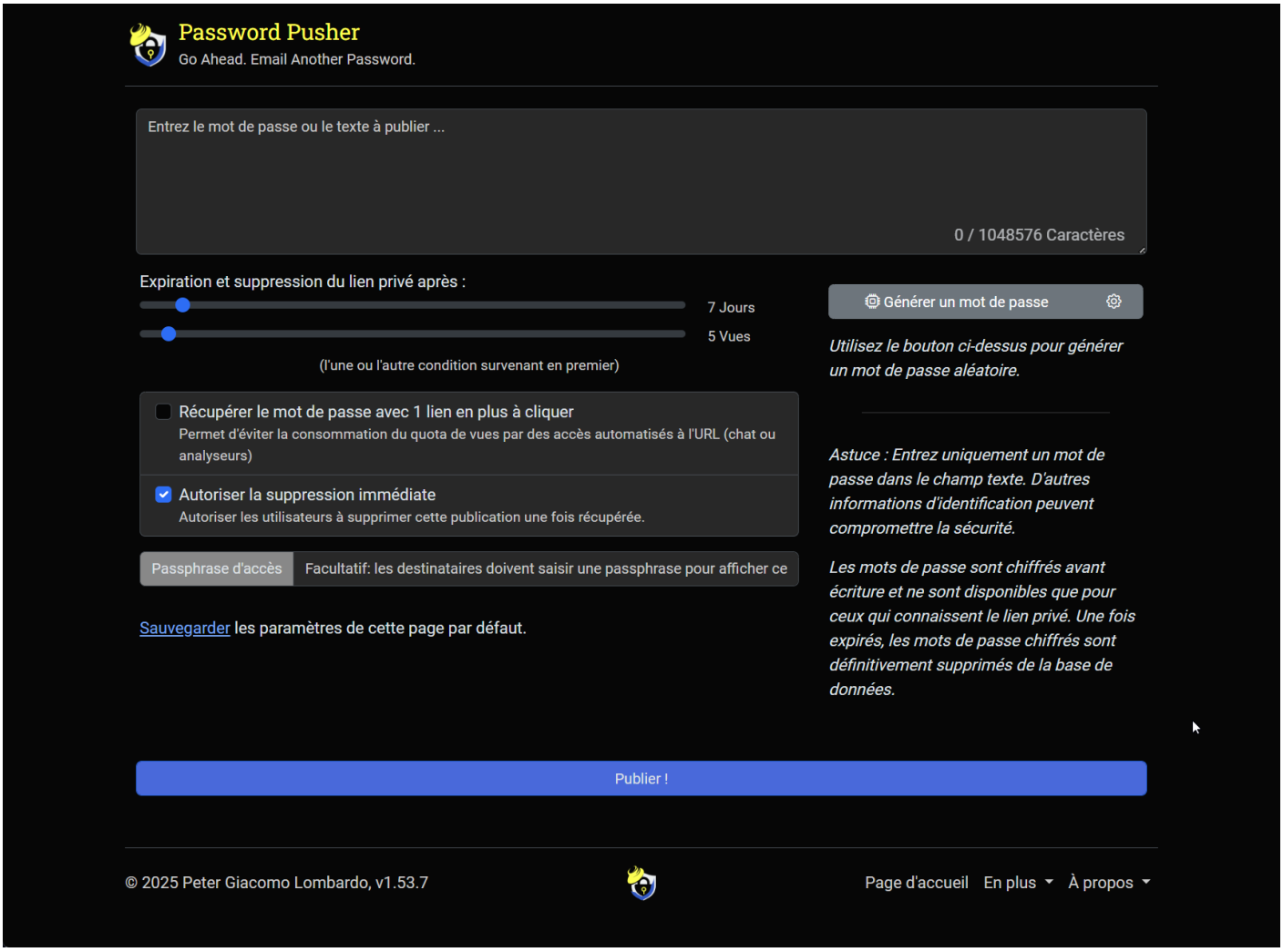
```
services:
  librespeed:
    image: lscr.io/linuxserver/librespeed:latest
    container_name: librespeed
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
#      - PASSWORD=PASSWORD
#      - CUSTOM_RESULTS=false #optional
#      - DB_TYPE=sqlite #optional
#      - DB_NAME=DB_NAME #optional
```

```
#      - DB_HOSTNAME=DB_HOSTNAME #optional
#      - DB_USERNAME=DB_USERNAME #optional
#      - DB_PASSWORD=DB_PASSWORD #optional
#      - DB_PORT=DB_PORT #optional
#      - IPINFO_APIKEY=ACCESS_TOKEN #optional
    volumes:
      - librespeed_config:/config
    ports:
      - 85:80
    restart: unless-stopped
networks:
  librespeed_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
volumes:
  librespeed_config:
    external: true
```

Compose

Password Pusher

Password Pusher est une application open source permettant de communiquer des informations sensibles sur le web. Les liens secrets expirent après un certain nombre de visualisations et/ou après un certain temps écoulé.



Le Docker Compose :

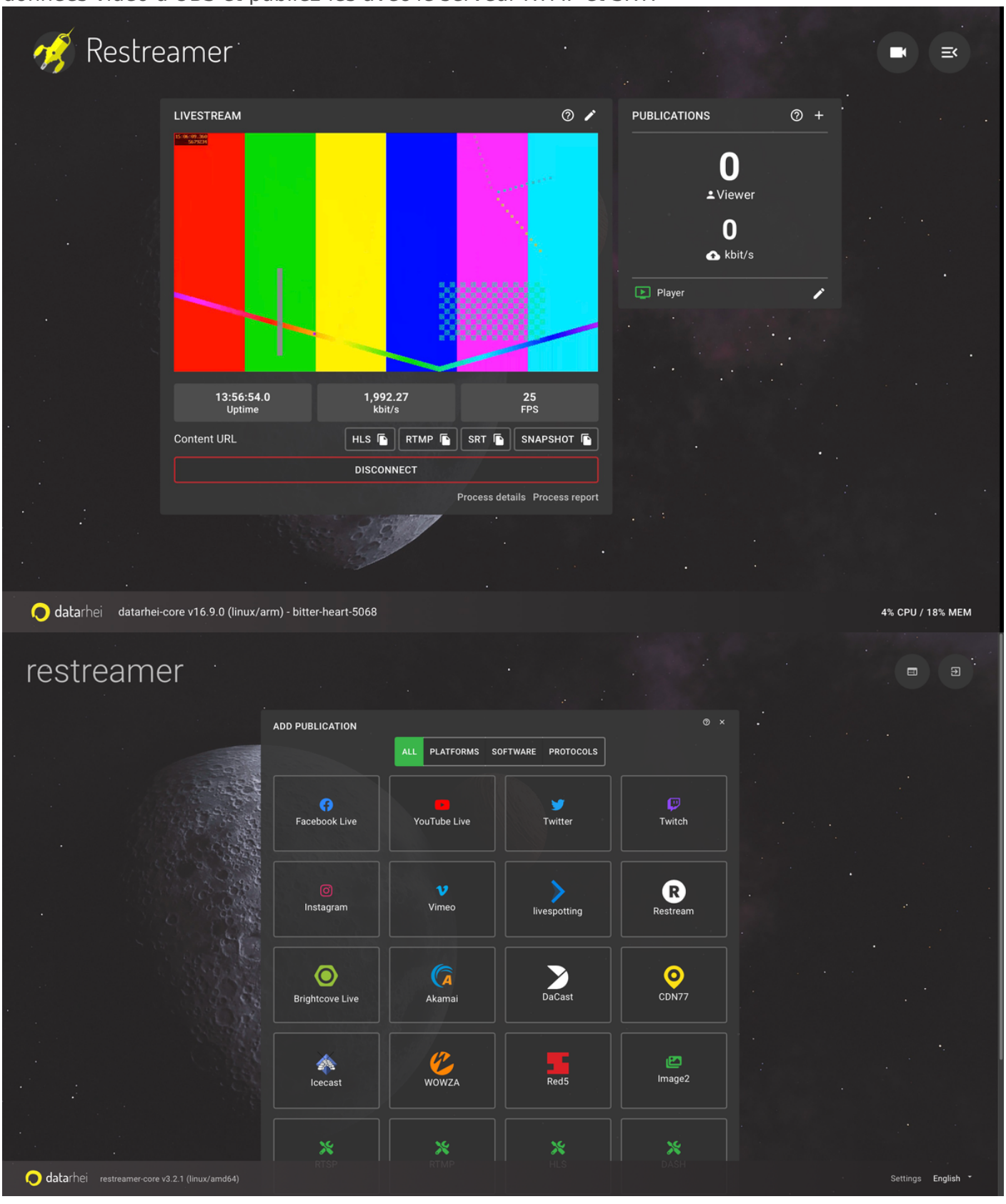
```
services:
  pwpush:
    container_name: passwordpusher
    image: pglombardo/pwpush:stable
    ports:
      - 5100:5100
    networks:
```

```
      - passwordpusher_network

networks:
  passwordpusher_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
```

Compose

Restreamer

Le Restreamer est une solution de serveur de diffusion en continu (streaming) complète pour l'auto-hébergement. Il dispose d'une interface utilisateur visuellement attrayante et ne comporte pas de frais de licence récurrents. Vous pouvez télécharger votre flux vidéo en direct sur YouTube, Twitch, Facebook, Vimeo ou d'autres solutions de diffusion en continu comme Wowza. Recevez les données vidéo d'OBS et publiez-les avec le serveur RTMP et SRT.



Crée des volumes :

```
docker volume create restreamer_data
docker volume create restreamer_config
```

Docker Compsose :

```
services:
    restreamer:
        image: 'datarhei/restreamer:latest'
        container_name: 'restreamer'
        ports:
#            - '6000:6000/udp'
#            - '1936:1936'
            - '1935:1935'
#            - '8181:8181'
            - '8080:8080'
        volumes:
            - restreamer_data:/core/data
            - restreamer_config:/core/config
        restart: always
networks:
  restreamer_network:
    driver: bridge
    # Permet la communication externe mais isole des autres conteneurs
    internal: false
volumes:
    restreamer_data:
      external: true
    restreamer_config:
      external: true
```

Documentation :

- https://docs.datarhei.com/restreamer

Compose

Jellyfin avec transcodage matériel NVIDIA

1. Installation de la boîte à outils NVIDIA Container
2. Faire un point de montage d'un partage de stockage réseau sur l'hôte, si ce n'est pas un NAS, en modifiant le fichier `fstab` sur Debian ou Ubuntu serveur avec les commandes :

```
sudo -i
```

```
nano /etc/fstab
```

```
192.168.0.187:/mnt/media/multimedia /mnt/nas_multimedia/ nfs defaults,nofail 0 0
```

```
apt install -y nfs-common && systemctl daemon-reload && mkdir /mnt/nas_multimedia/ && mount -a
```

```
exit
```

3. Crée un volume pour le conteneur :

```
docker volume create jellyfin_config
```

```
docker volume create jellyfin_cache
```

4. Le docker compose :

```
services:
  jellyfin:
    image: 'linuxserver/jellyfin'
    container_name: 'Jellyfin'
    network_mode: 'bridge'
    restart: 'unless-stopped'
    hostname: 'Jellyfin'
    environment:
      TZ: 'Europe/Zurich'
      PUID: '0'
```

```
      PGID: '0'
      JELLYFIN_PublishedServerUrl: '0.0.0.0'
      NVIDIA_DRIVER_CAPABILITIES: 'all'
      NVIDIA_VISIBLE_DEVICES: 'all'
    deploy:
      resources:
        limits:
          cpus: '2.00'
          memory: '1024M'
        reservations:
          cpus: '0.02'
          memory: '256M'
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
    volumes:
      - 'jellyfin_config:/config'
      - 'jellyfin_cache:/cache'
      - '/mnt/nas_multimedia/Films:/films'
      - '/mnt/nas_multimedia/Séries:/tv_shows'
    ports:
      - '8096:8096'
networks:
  jellyfin_network:
    driver: bridge
volumes:
  jellyfin_config:
    external: true
  jellyfin_cache:
    external: true
```

Voir les sessions d'encodage sur l'hôte :

```
nvidia-smi encodersessions
```

Pour voir l'utilisation en temps réel de façon globale de la carte graphique :

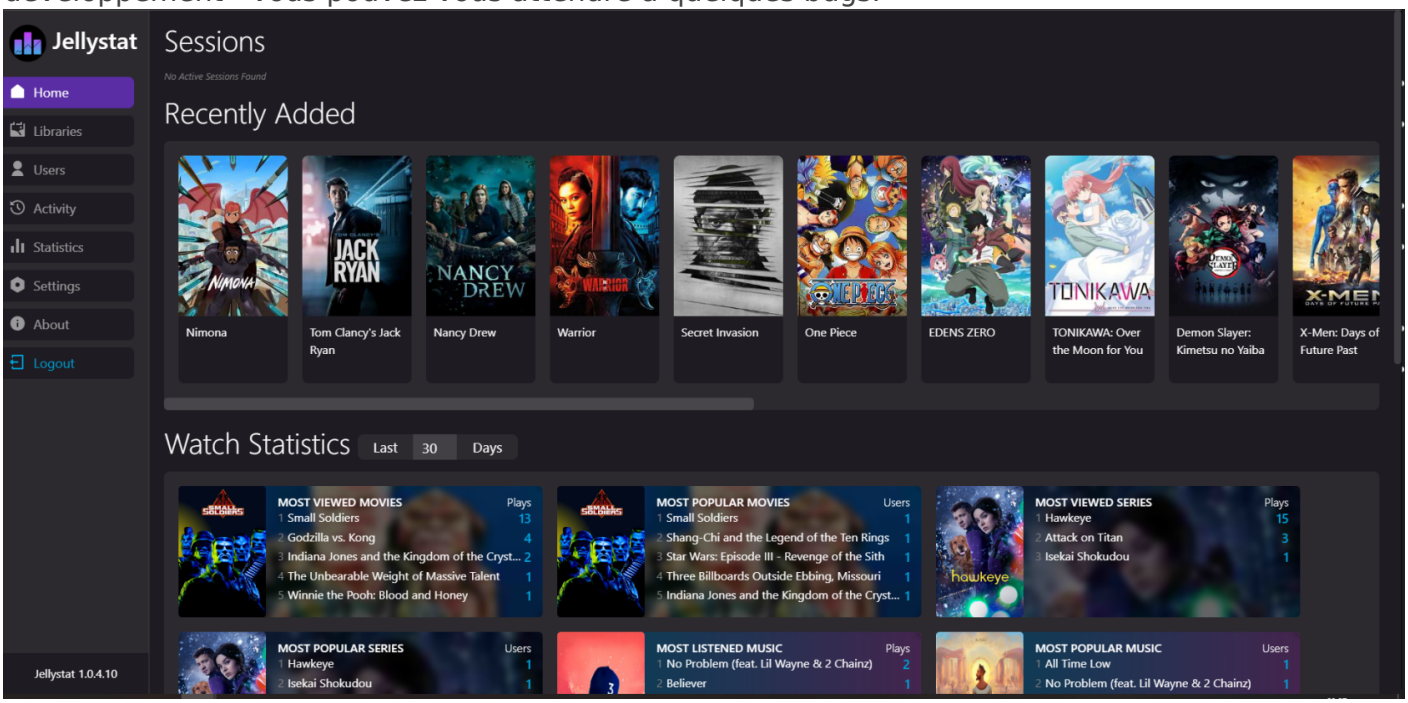
```
nvidia-smi dmon
```

Documentation : https://hub.docker.com/r/linuxserver/jellyfin

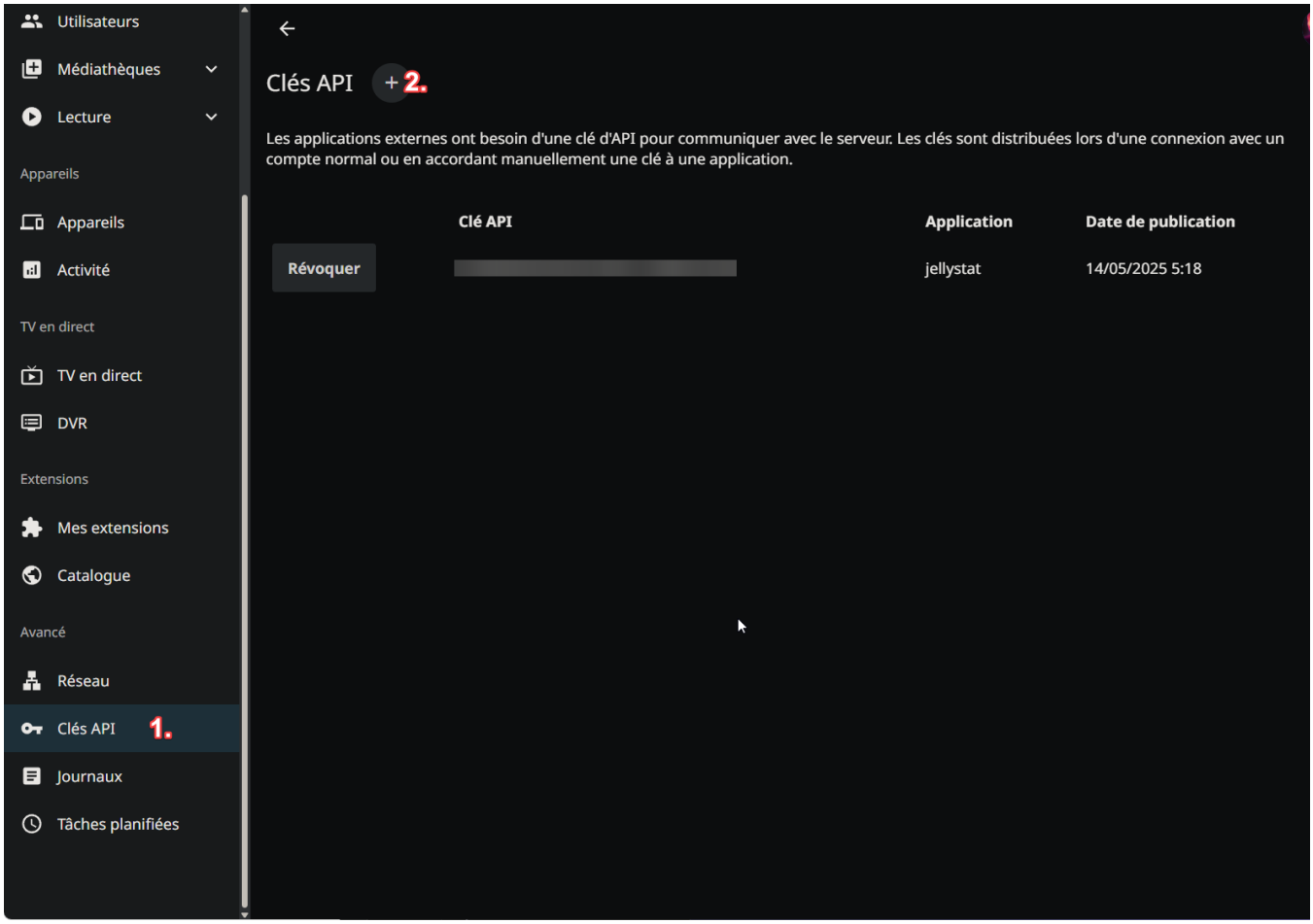
Compose

JellyStat

Application statistique gratuite et open source pour Jellyfin ! Ce projet est encore en développement - vous pouvez vous attendre à quelques bugs.



1. Demandez la clé API :



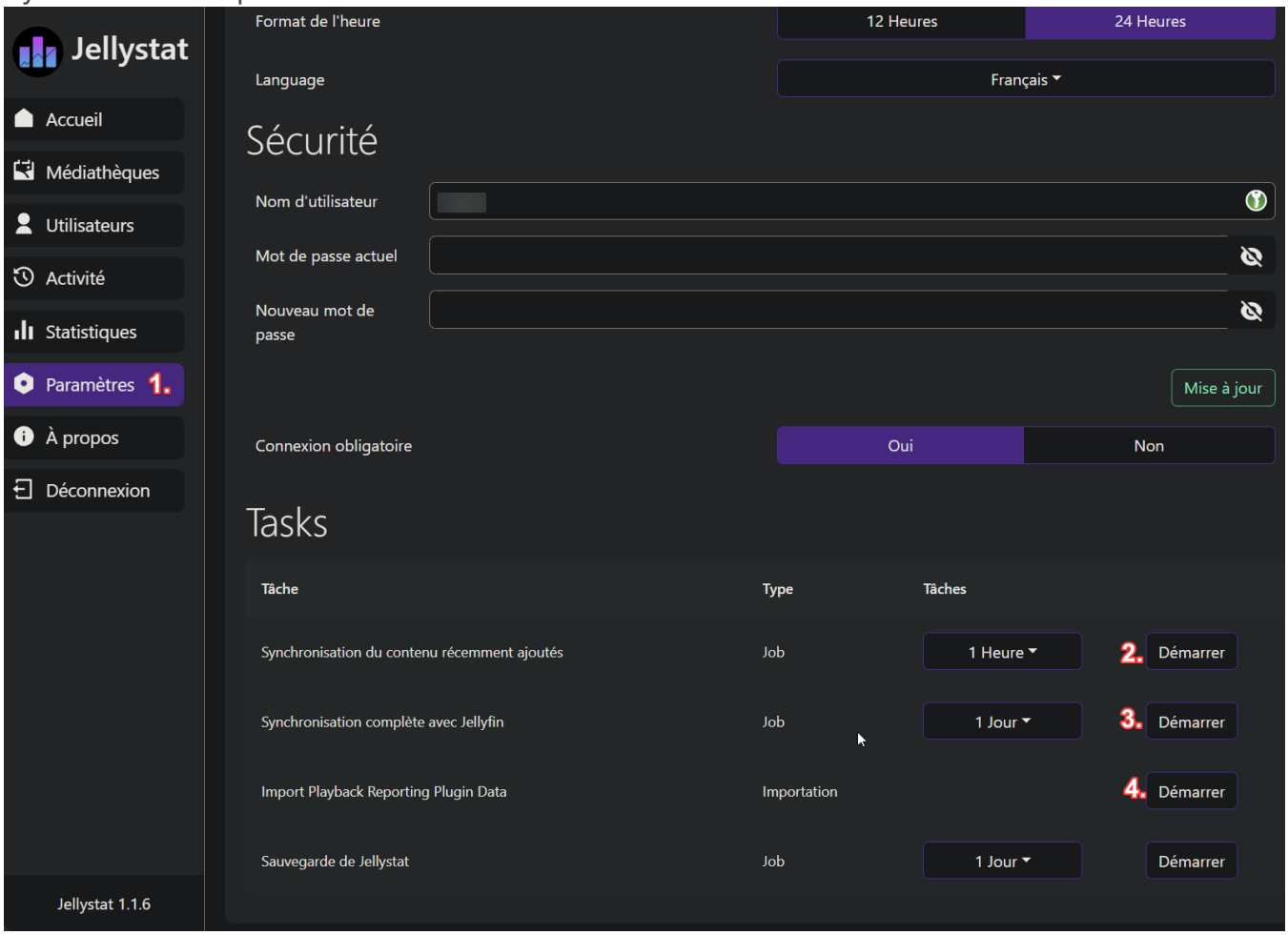

```
docker volume create jellystat_db
docker volume create jellystat_backup
```

```
services:
  jellystat-db:
    image: postgres:15.2
    restart: unless-stopped
    container_name: 'jellystat_db'
    environment:
      POSTGRES_DB: 'jfstat'
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: Mik4-T3-Mub7 # Changez le mot de passe
    volumes:
      - jellystat_db:/var/lib/postgresql/data

  jellystat:
    image: cyfershepard/jellystat:latest
    container_name: 'jellystat'
    environment:
```

```
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: Mik4-T3-Mub7 # Changez le mot de passe
      POSTGRES_IP: jellystat-db
      POSTGRES_PORT: 5432
      JWT_SECRET: 'Kyd8-Buk4=F3' # Changez le mot de passe
    ports:
      - "2121:3000"
    volumes:
      - jellystat_backup:/app/backend/backup-data
    depends_on:
      - jellystat-db
    restart: unless-stopped

volumes:
  jellystat_db:
    external: true
  jellystat_backup:
    external: true
```

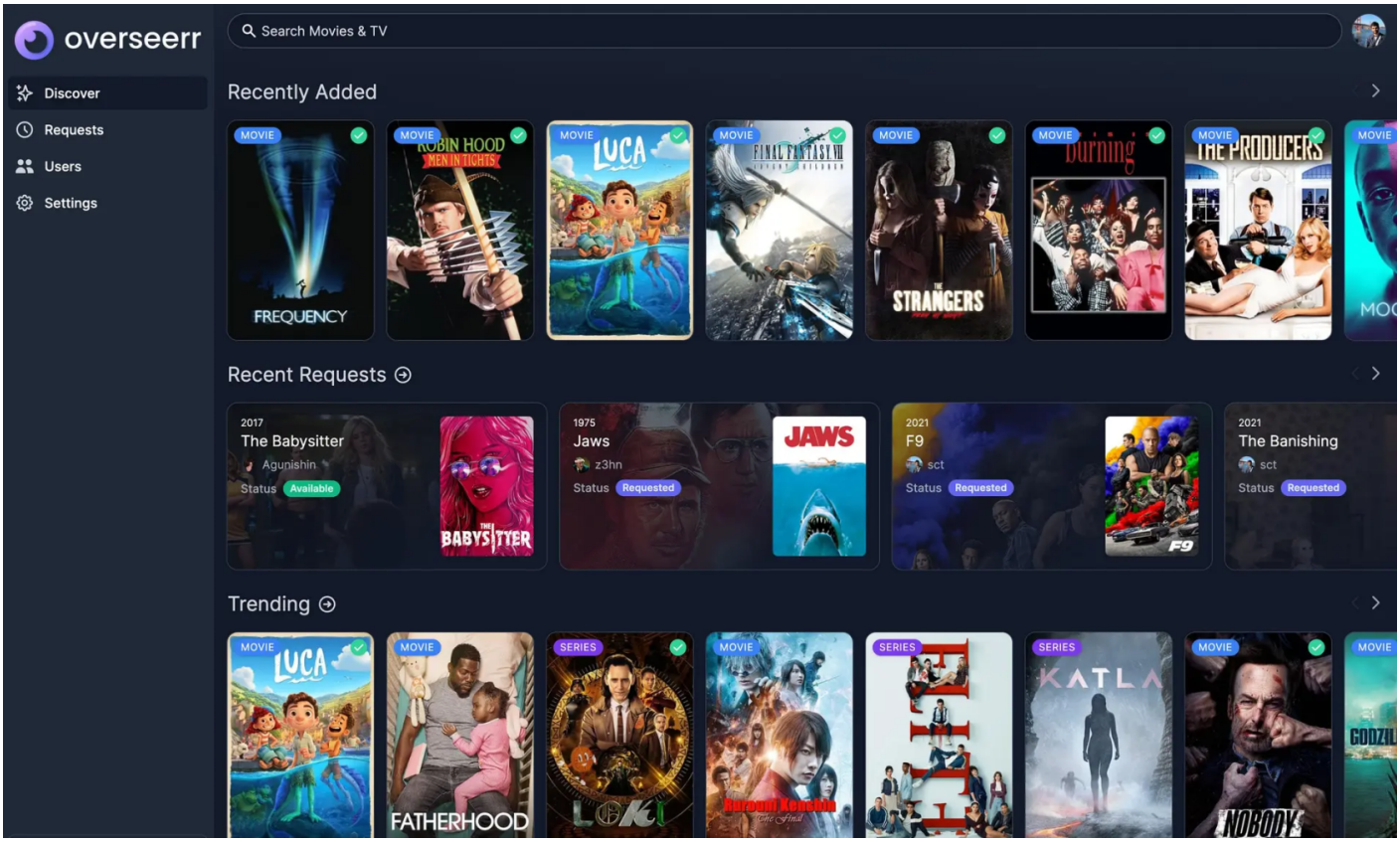
2. Accédez à la page administration `ip:2121` et créez-vous un compte
3. Insérez la clé API et lien de la page Jellyfin

4. Synchronisez les paramètres :



Compose

Jellyseerr




```
docker volume create jellyseerr_config
```

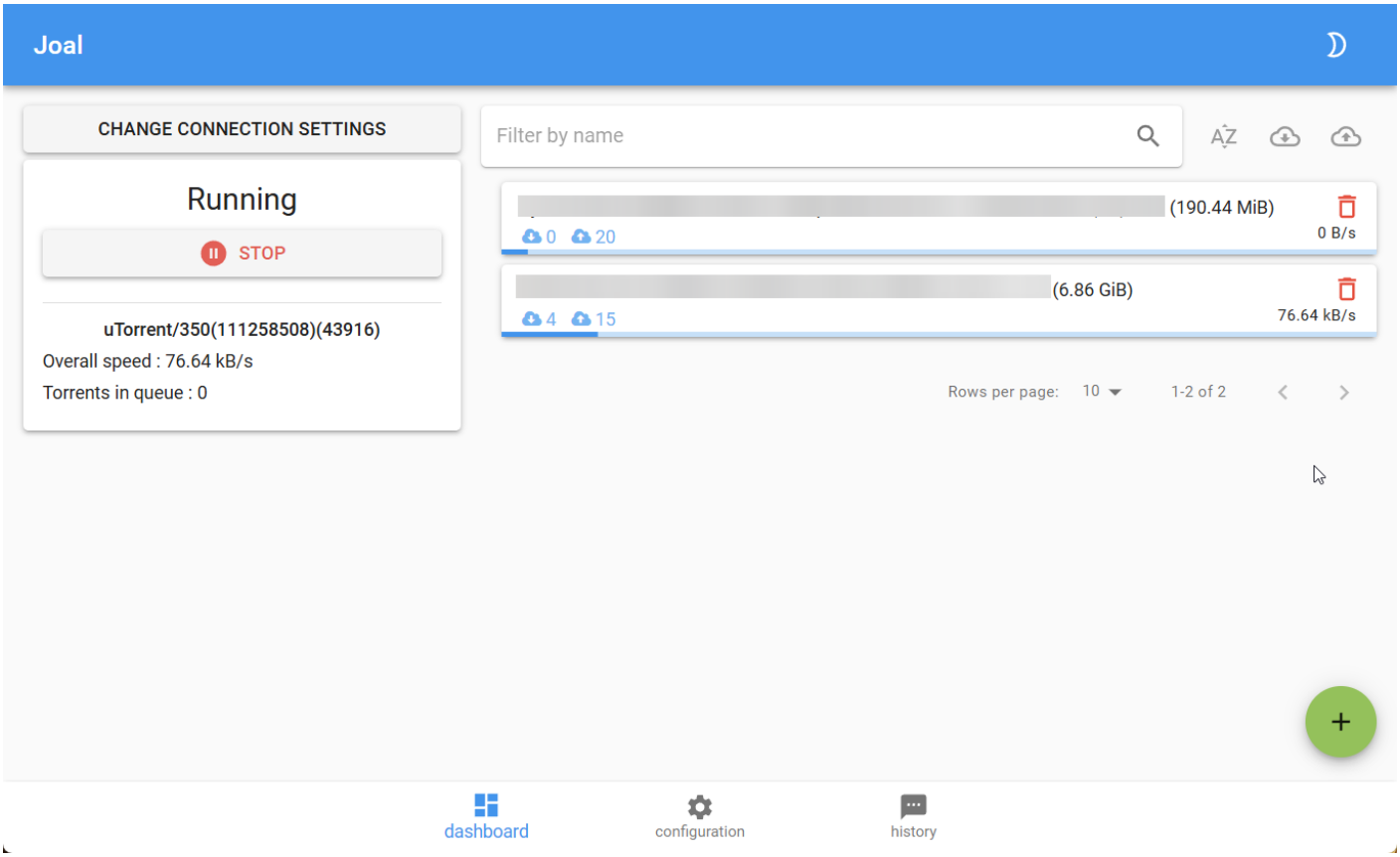
```
services:
  jellyseerr:
    image: ghcr.io/fallenbagel/jellyseerr:latest
    restart: unless-stopped
    container_name: jellyseerr
    environment:
      - LOG_LEVEL=debug
      - TZ=Europe/Zurich
      - PORT=5055 #optional
    ports:
      - 5055:5055
    volumes:
      - jellyseerr_config:/app/config
```

```
volumes:
  jellyseerr_config:
    external: true
```

Compose

Joal

Il fait croire aux trackers torrent que vous partagez des fichiers pour augmenter votre ratio et tricher.



Adaptez les dossiers de destination, dans l'exemple, utilisez votre dossier à la racine que vous avez déjà créé auparavant `/srv` :

```
mkdir -p /srv/joal/data
cd /srv/joal
wget https://github.com/anthonyraymond/joal/releases/download/2.1.36/joal.tar.gz
tar -xvzf joal.tar.gz -C ./data
rm joal.tar.gz
```

Le docker compose :

```
services:
  joal:
    image: anthonyraymond/joal:latest
```

```
    container_name: joal
    restart: unless-stopped
    volumes:
      - /srv/joal/data:/data
    ports:
      - 6060:8080
    command:
      - "--joal-conf=/data"
      - "--spring.main.web-environment=true"
      - "--server.port=8080"
      - "--joal.ui.path.prefix=joal"
      - "--joal.ui.secret-token=YourToken"
```

Avertissements :

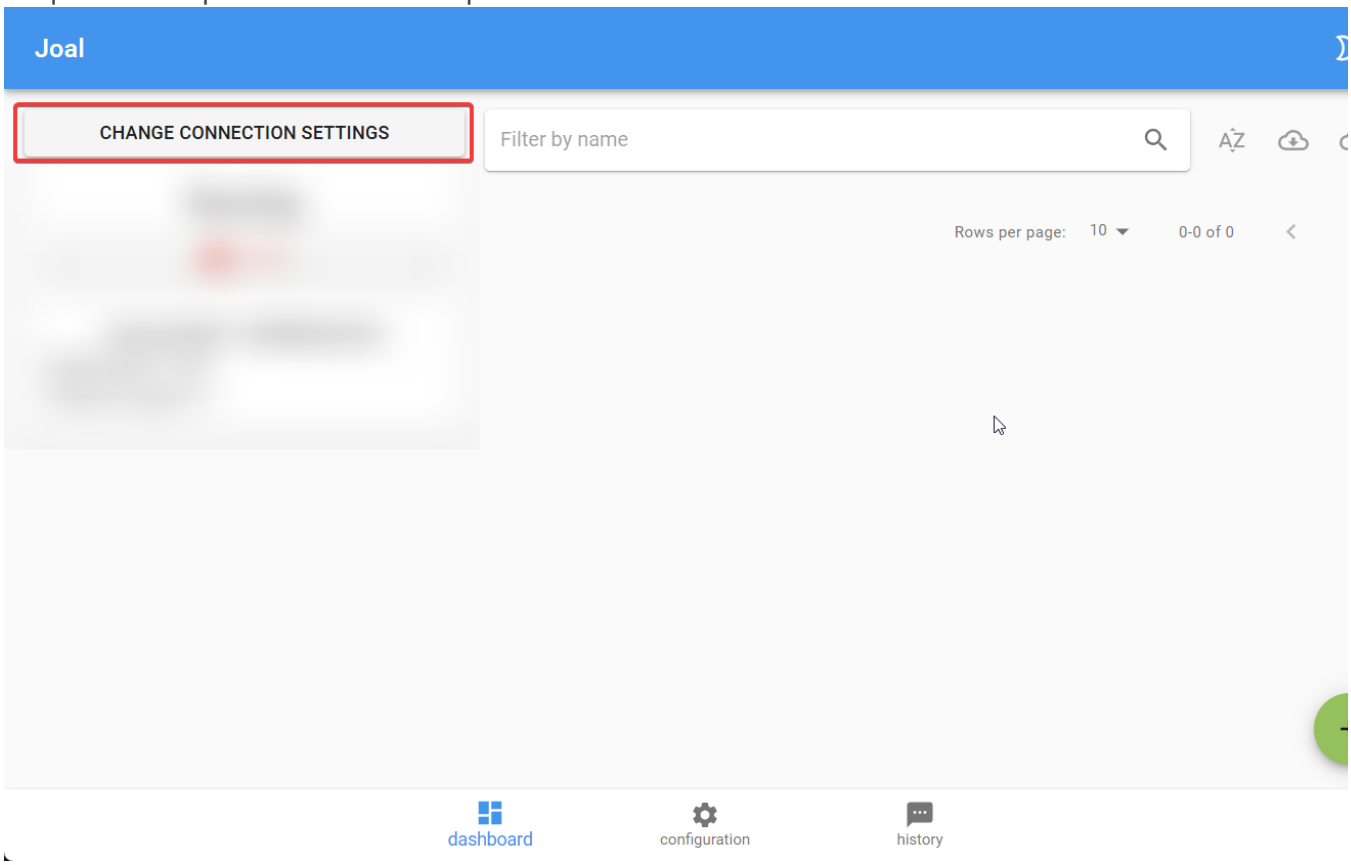
- ⚠ Mettez un mot de passe fort pour `YourToken` et ne pas exposer au public la page d'administration joal
- ⚠ Si vous êtes satisfait de votre ratio et voulez éviter de vous faire bannir, stoppez le simulateur en ajoutant un `#` à `restart: unless-stopped` dans le fichier compose.yml. Cela empêchera le conteneur de redémarrer automatiquement, car le simulateur lance l'upload au démarrage.

Accédez à l'interface web, il est important de mettre `/joale/ui` :

```
http://ip:6060/joal/ui
```

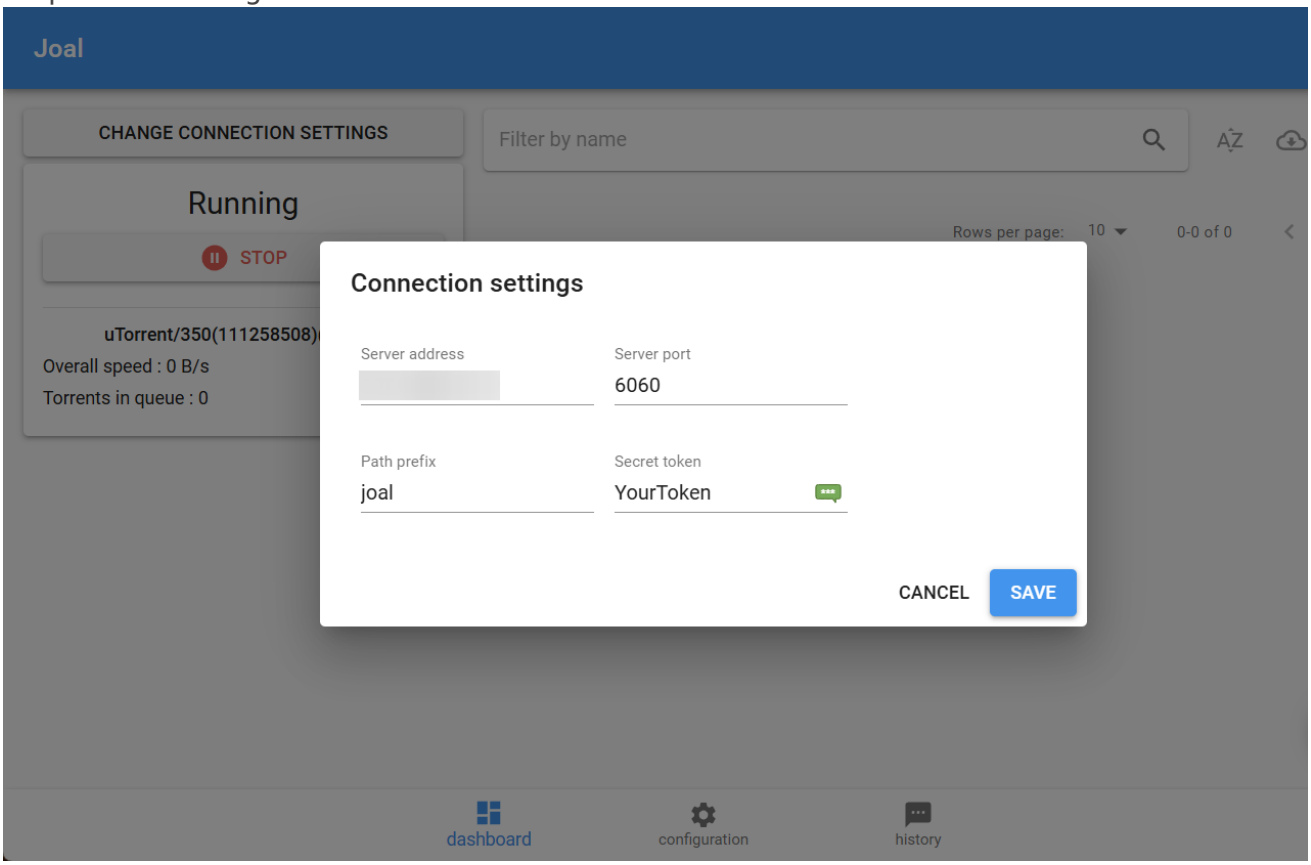
Étapes d'utilisation :

1. Cliquez sur l'option "Modifier les paramètres de connexion".



2. Lors de la première connexion, configurez les paramètres de connexion :
   - Adresse du serveur : saisissez l'adresse IP du serveur
   - Port du serveur : 6060
   - Préfixe du chemin : joal
   - Jeton secret : entrez le jeton choisi, ici "YourToken"

- Cliquez sur "Enregistrer"

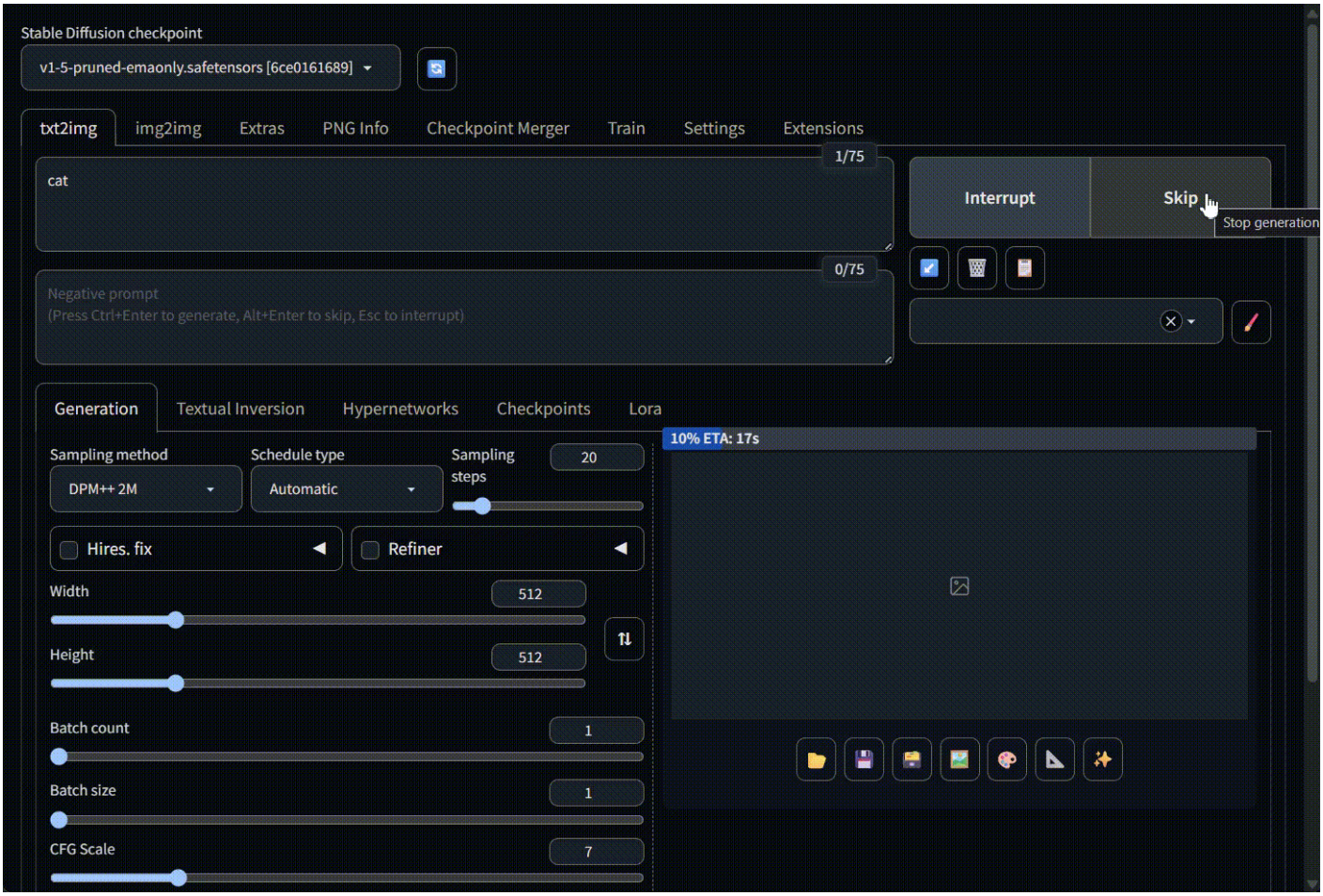


3. L'onglet "Configuration" vous permet de paramétrer différentes options :
   - Vitesse minimale et maximale de téléchargement
   - Votre client de torrent préféré
   - Nombre maximum de torrents pouvant être envoyés simultanément
   - Ratio requis pour chaque torrent (utilisez "-1" pour un ratio illimité)
4. Il est recommandé d'être prudent dans les paramètres de configuration et de ne pas trop augmenter les vitesses de téléchargement.
5. Il existe trois façons d'ajouter un torrent à Joal :
   - Glisser-déposer les torrents dans le tableau de bord
   - Cliquer sur "+" et sélectionner le torrent
   - Placer le torrent dans le dossier "joal/data/torrents"
6. Il est préférable de privilégier les torrents qui ont beaucoup de seeds pour que le partage fonctionne.

Compose

Stable Diffusion web UI NVIDIA

Stable Diffusion web UI est un projet open-source de génération d'images. Il permet de créer des images réalistes et créatives à partir de texte. Il a connu un essor important ces dernières années, grâce à sa capacité à générer des images très détaillées et variées, et à sa flexibilité grâce à sa communauté active et à ses nombreux outils.



1. Installation de la boîte à outils NVIDIA Container
2. Créer les répertoires et lancer Docker Compose :

```
mkdir -p /home/docker/srv/stable-
diffusion/{inputs,templates,embeddings,extensions,models,localizations,outputs}
chown -R 1000:1000 /home/docker/srv/stable-diffusion
```

```
services:
  stable-diffusion-webui:
    image: universonic/stable-diffusion-webui:minimal
```

```
    command: --no-half --no-half-vae --precision full
    runtime: nvidia
    restart: unless-stopped
    ports:
      - "8080:8080/tcp"
    environment:
      TZ: 'Europe/Zurich'
      PUID: '1000'
      PGID: '1000'
    volumes:
      - /home/docker/srv/stable-diffusion/inputs:/app/stable-diffusion-webui/inputs
      - /home/docker/srv/stable-diffusion/templates:/app/stable-diffusion-
webui/textual_inversion_templates
      - /home/docker/srv/stable-diffusion/embeddings:/app/stable-diffusion-webui/embeddings
      - /home/docker/srv/stable-diffusion/extensions:/app/stable-diffusion-webui/extensions
      - /home/docker/srv/stable-diffusion/models:/app/stable-diffusion-webui/models
      - /home/docker/srv/stable-diffusion/localizations:/app/stable-diffusion-
webui/localizations
      - /home/docker/srv/stable-diffusion/outputs:/app/stable-diffusion-webui/outputs
    cap_drop:
      - ALL
    cap_add:
      - NET_BIND_SERVICE
    deploy:
      mode: global
      placement:
        constraints:
          - "node.labels.iface != extern"
      restart_policy:
        condition: unless-stopped
      resources:
        reservations:
          devices:
            - driver: nvidia
              capabilities: [gpu]
```

Documentation :

- https://hub.docker.com/r/universonic/stable-diffusion-webui

Compose

Exemple de documentation : NomDuService

NomDuService est un service dont la fonction principale est de [décrire brièvement le rôle du service, ex. : générer des images, traiter des données, héberger une base, etc.].

1. Prérequis

- Docker >= 20.x
- NVIDIA Container Toolkit, si le conteneur exploite le GPU (optionnel)

2. Arborescence des répertoires

Les volumes suivants sont utilisés afin de persister les données :

```
mkdir -p /home/docker/srv/nomduservice/{entrees,modeles,resultats}
chown -R 1000:1000 /home/docker/srv/nomduservice
```

Répertoire hôte	Répertoire dans le conteneur	Description
`/home/docker/srv/nomduservice/entrees`	`/app/entrees`	Fichiers d’entrée
`/home/docker/srv/nomduservice/modeles`	`/app/modeles`	Modèles utilisés
`/home/docker/srv/nomduservice/resultats`	`/app/resultats`	Résultats générés

3. Exemple de configuration `docker-compose.yml`

```
services:
  nomduservice:
    image: exemple/image:latest
```

```
    container_name: nomduservice
    ports:
      - "8080:8080"
    environment:
      TZ: "Europe/Paris"
      PUID: "1000"
      PGID: "1000"
    restart: unless-stopped
    volumes:
      - /home/docker/srv/nomduservice/entrees:/app/entrees
      - /home/docker/srv/nomduservice/modeles:/app/modeles
      - /home/docker/srv/nomduservice/resultats:/app/resultats
```

4. Accès

- URL d’accès local : `http://localhost:8080` (à adapter selon l’environnement)

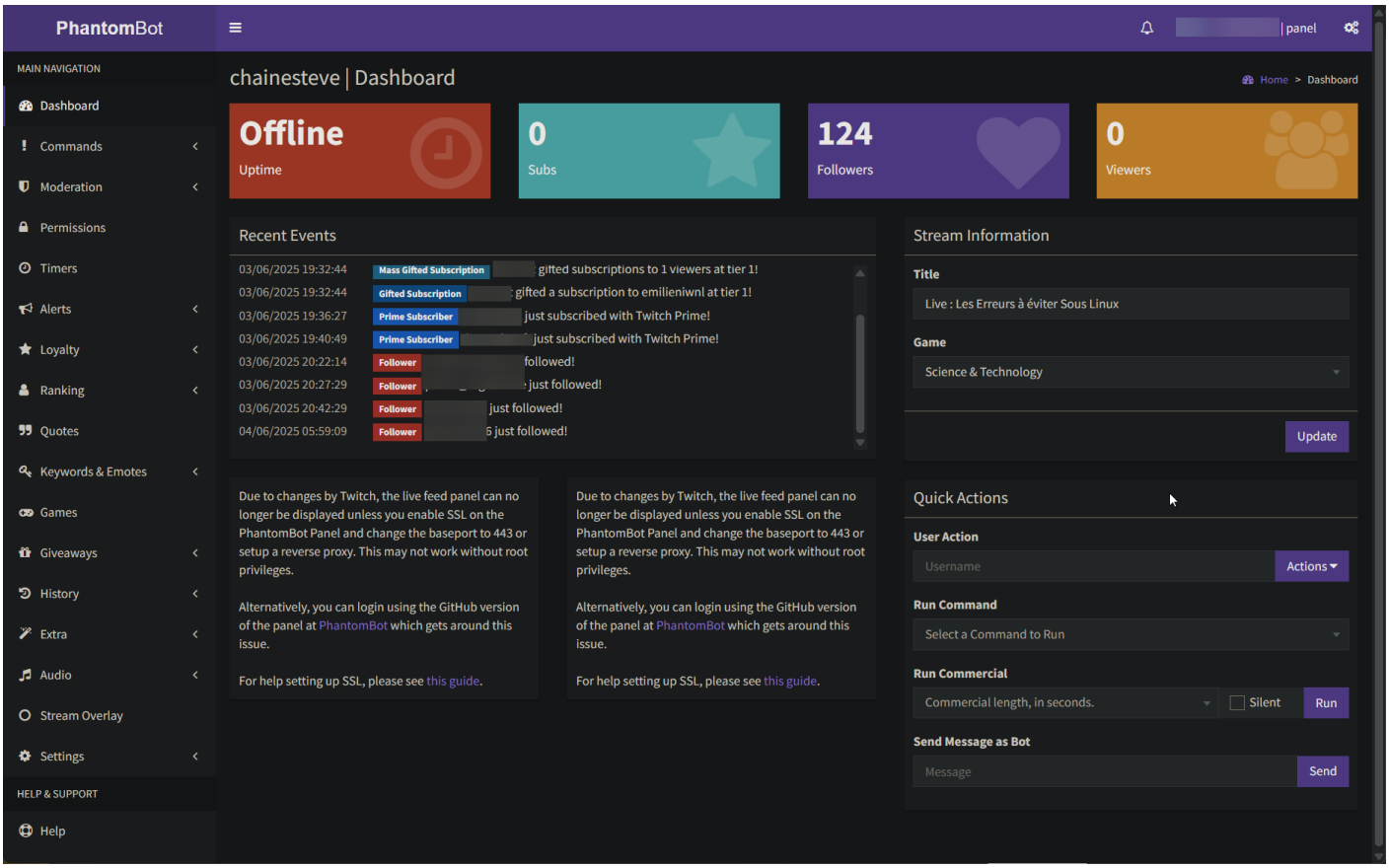
5. Documentation complémentaire

- Image Docker : *à renseigner*
- Référentiel Git : *à renseigner*
- Documentation officielle : *à renseigner*

Compose

PhantomBot

PhantomBot est un bot open source polyvalent conçu principalement pour les streamers sur des plateformes telles que Twitch et YouTube. Il offre une large gamme de fonctionnalités pour automatiser et enrichir l'interaction avec les spectateurs, comme la gestion de commandes personnalisées, la modération de chat, les alertes, les sondages, et bien plus encore. Grâce à son interface web intuitive et à sa grande flexibilité, PhantomBot permet aux créateurs de contenu de personnaliser leur expérience de streaming selon leurs besoins, tout en facilitant l'engagement de leur communauté.



Prérequis :

- Debian 12 avec le compte root désactivé déjà installé avec des mises automatiques de sécurité
- Docker Rootless pour de la sécurité avec Portainer avec plus de 500 templates !

1. Préparez le dossier `srv`, il contiendra tous les dossiers de vos applications : Création du dossier de destination des données PhantomBot :

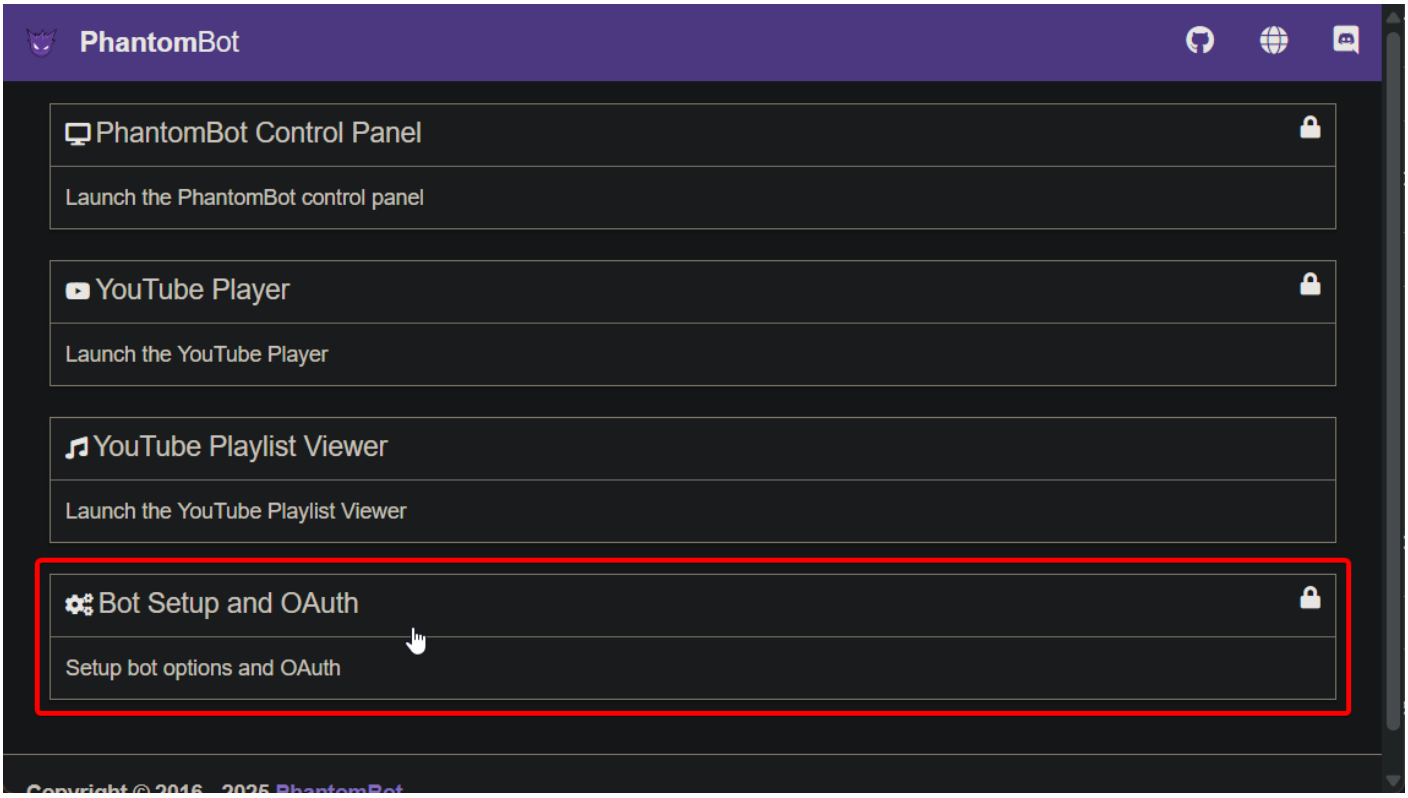
```
mkdir -p $HOME/srv/phantombot_data
```

2. Docker compose. Remplacez `nomdutilisateur` par votre nom d'utilisateur :

```
services:
  phantombot:
    container_name: phantombot
    image: ghcr.io/phantombot/phantombot
    ports:
      - target: 25000
        published: 25000
        protocol: tcp
    restart: unless-stopped
    volumes:
      - /home/nomdutilisateur/srv/phantombot_data:/opt/PhantomBot_data
```

3. Regardez les logs, ils donnent les informations de connexion, et rendez-vous sur la page de PhantomBot :

```
ip:25000
```



Sauvegarde de `/srv/phantombot_data` :

1. Création du dossier de destination des sauvegardes :

```
mkdir -p $HOME/srv/sauvegarde_conteneur_docker
```

2. Script de sauvegarde :

```
nano sauvegarde_conteneur_docker.sh
```

```
#!/bin/bash
# Description
# Ce script effectue une sauvegarde des dossiers spécifiés.
# Dossiers à sauvegarder
BACKUP_DIRS=("$HOME/srv/phantombot_data")
# Destination des sauvegardes
BACKUP_DEST="$HOME/srv/sauvegarde_conteneur_docker"
# Exécution de la sauvegarde
for dir in "${BACKUP_DIRS[@]}"; do
    filename="$(basename "$dir")_$(date +%Y-%m-%d).tar"
    tar -cf "$BACKUP_DEST/$filename" "$dir"
    # Création du fichier de checksum SHA-256
    sha256sum "$BACKUP_DEST/$filename" > "$BACKUP_DEST/$filename.sha256"
done
# Fin du script
echo "Sauvegarde terminée."
# Pour vérifier le checksum d'un fichier de sauvegarde, utilisez la commande suivante :
# sha256sum -c "$BACKUP_DEST/nom_du_fichier.tar.sha256"
# Remplacez "nom_du_fichier" par le nom de votre fichier de sauvegarde.
```

Exécution du script

1. Avant de pouvoir exécuter le script, il faut lui donner les droits d'exécution :

```
chmod +x sauvegarde_conteneur_docker.sh
```

2. Pour lancer manuellement le script, utilisez la commande suivante depuis le dossier où se trouve le script :

```
./sauvegarde_conteneur_docker.sh
```

Ajout d'un cron pour une sauvegarde hebdomadaire

1. Pour automatiser la sauvegarde une fois par semaine, ouvrez le crontab de l'utilisateur avec :

```
crontab -e
```

2. Ajoutez la ligne suivante pour exécuter le script chaque dimanche à 2h du matin (modifiez le chemin vers le script si besoin) :

```
0 2 * * 0 /chemin/vers/sauvegarde_conteneur_docker
```

Remplacez `/chemin/vers/sauvegarde_conteneur_docker.sh` par le chemin absolu vers votre script.

3. Sauvegardez et fermez l'éditeur.

La tâche cron effectuera désormais une sauvegarde hebdomadaire automatiquement.

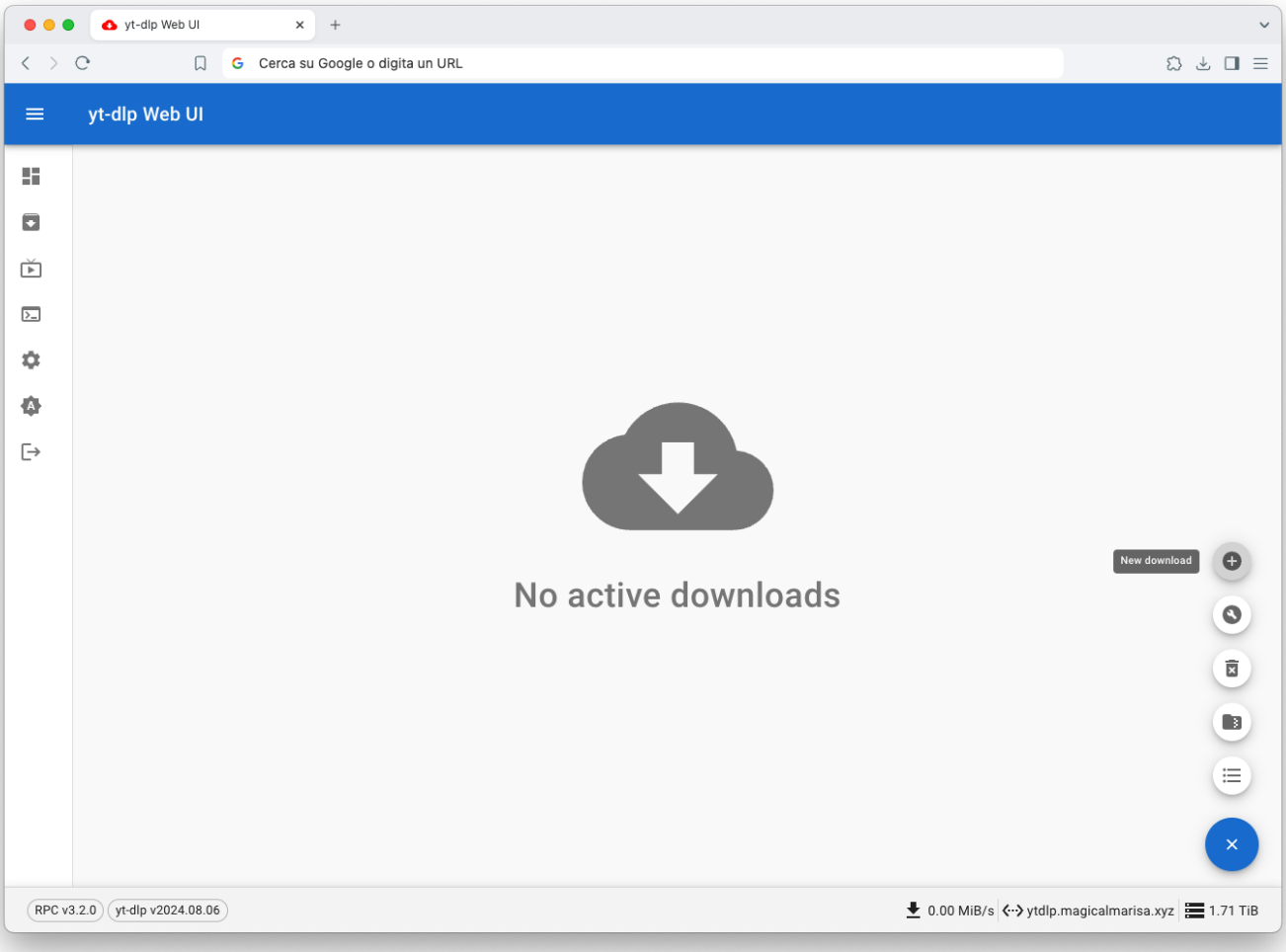
Documentation :

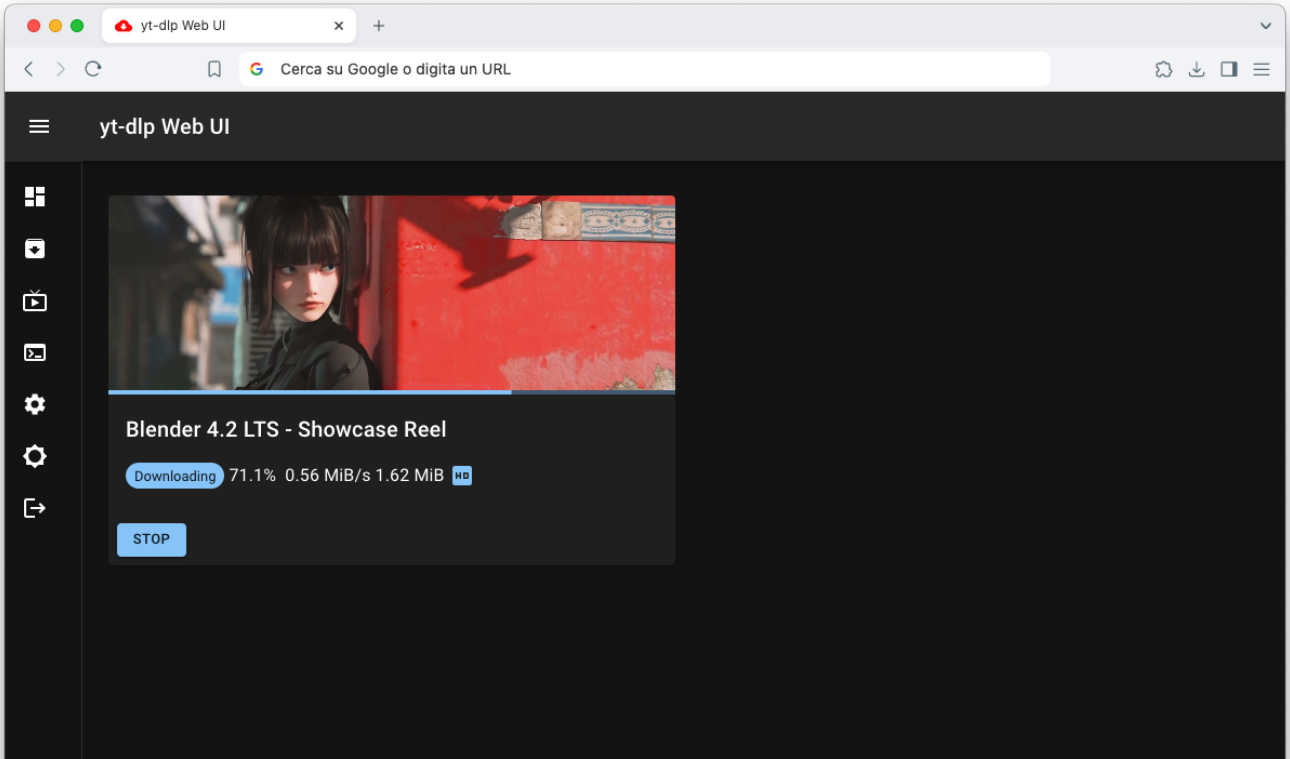
- https://phantombot.dev/guides/#guide=content-stable/guides&channel=stable

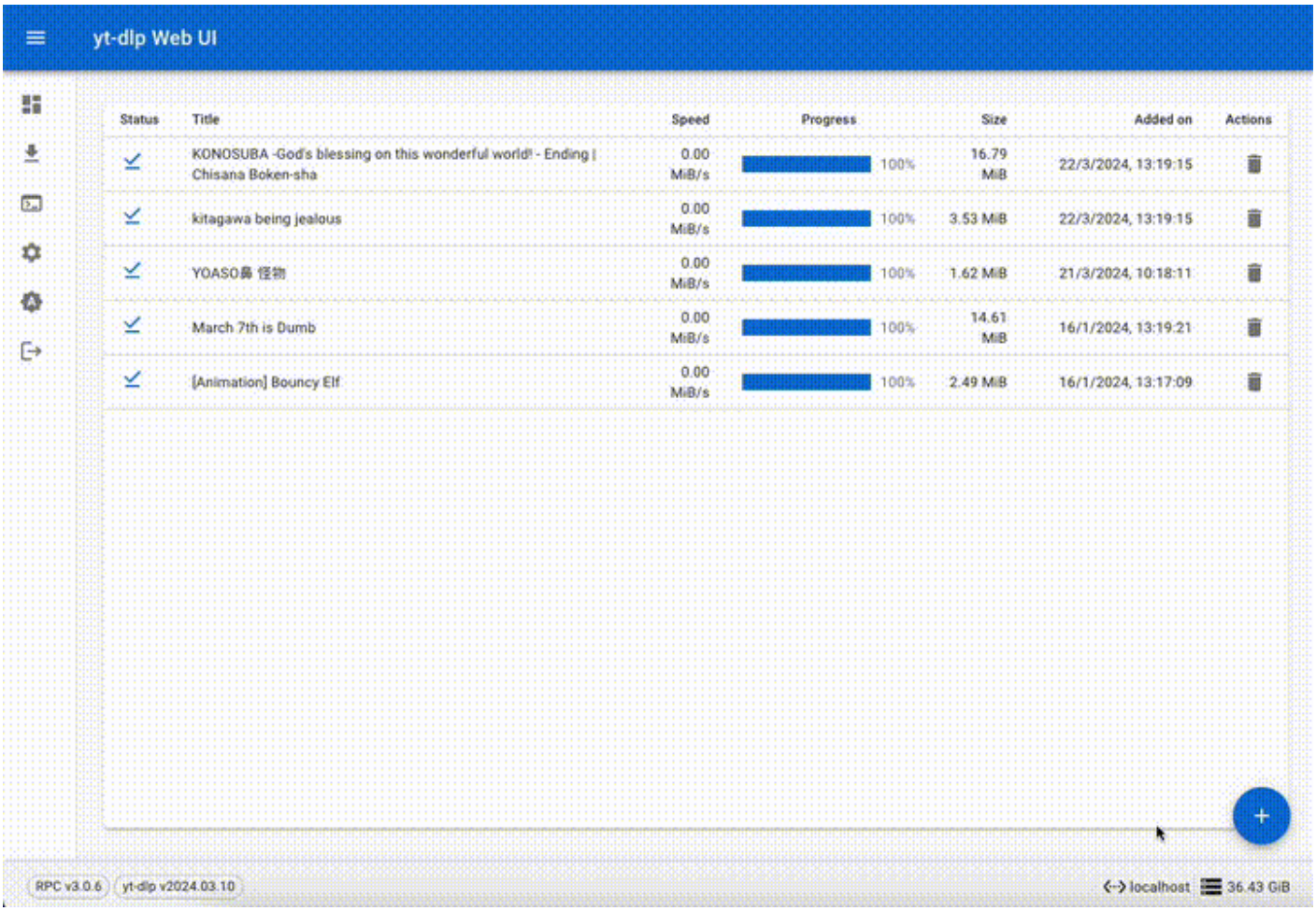
Compose

yt-dlp Web UI

Interface simple pour télécharger facilement des vidéos YouTube, Twitch, X.com, etc.







1. Docker Compose

```
services:
  yt-dlp-webui:
      image: marcobaobao/yt-dlp-webui
      container_name: yt-dlp-webui
      restart: unless-stopped
      ports:
          - 3033:3033
      volumes:
          - /chemindestockage/yt-dlp-webui:/downloads  # Changez le chemin du dossier
```

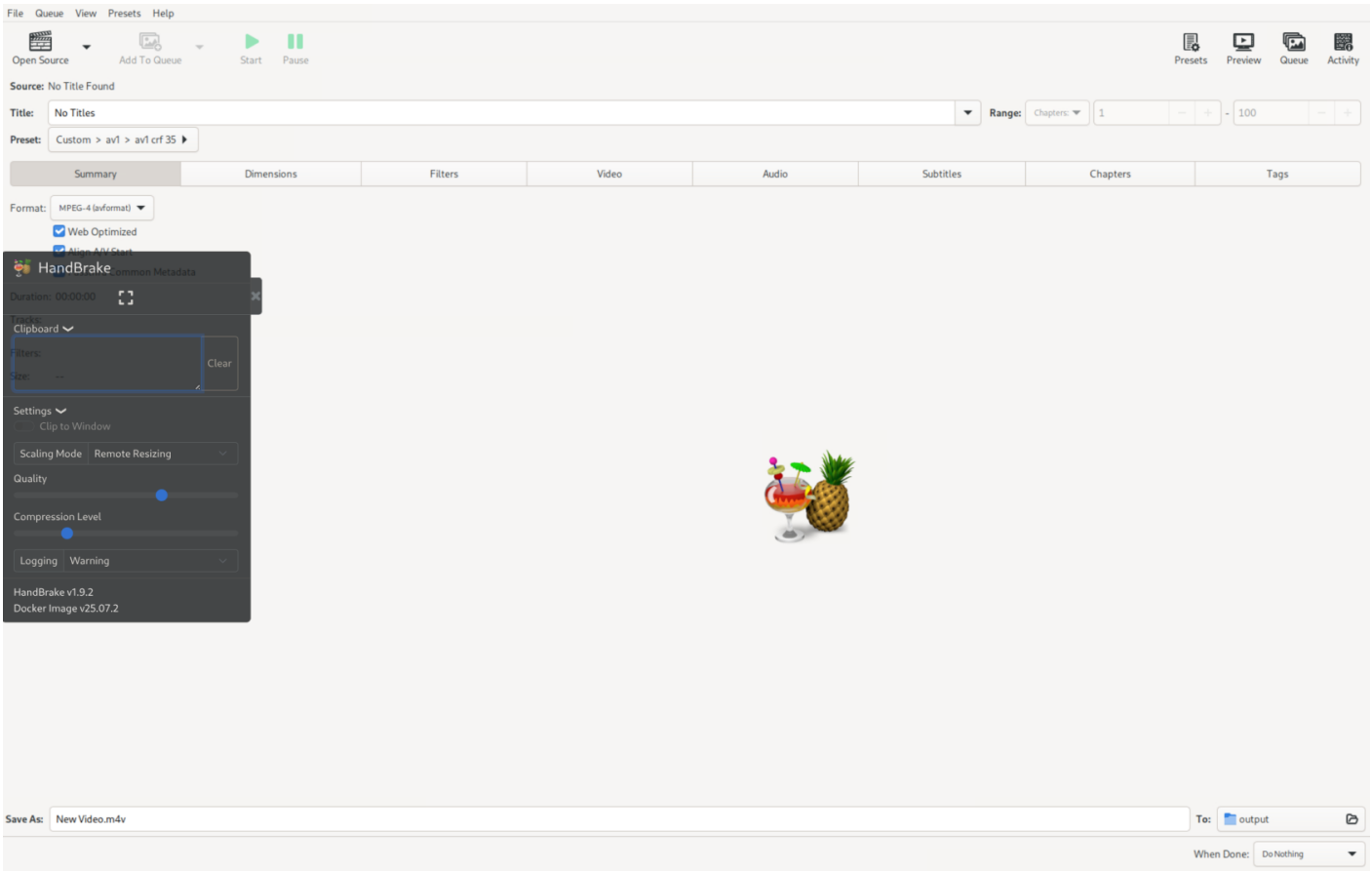
2. Accéder à l’interface web

```
http://<ip_serveur>:3033
```

Compose

handbrake

C'est un logiciel de conversion vidéo facile à utiliser, permettant de convertir des fichiers dans différents formats.



1. Docker Compose

```
services:
    handbrake:
        container_name: handbrake
        image: jlesage/handbrake
        restart: unless-stopped
        environment:
            - PUID=65534
            - PGID=65534
        ports:
```

```
            - 5800:5800
        volumes:
            - /home/docker/srv/handbrake/config:/config    # Changez le chemin du dossier
            - /mnt/nas_multimedia/handbrake/storage        # Changez le chemin du dossier
            - /mnt/nas_multimedia/handbrake/watch:/watch   # Changez le chemin du dossier
            - /mnt/nas_multimedia/handbrake/output:/output # Changez le chemin du dossier
```

2. Accéder à l’interface web

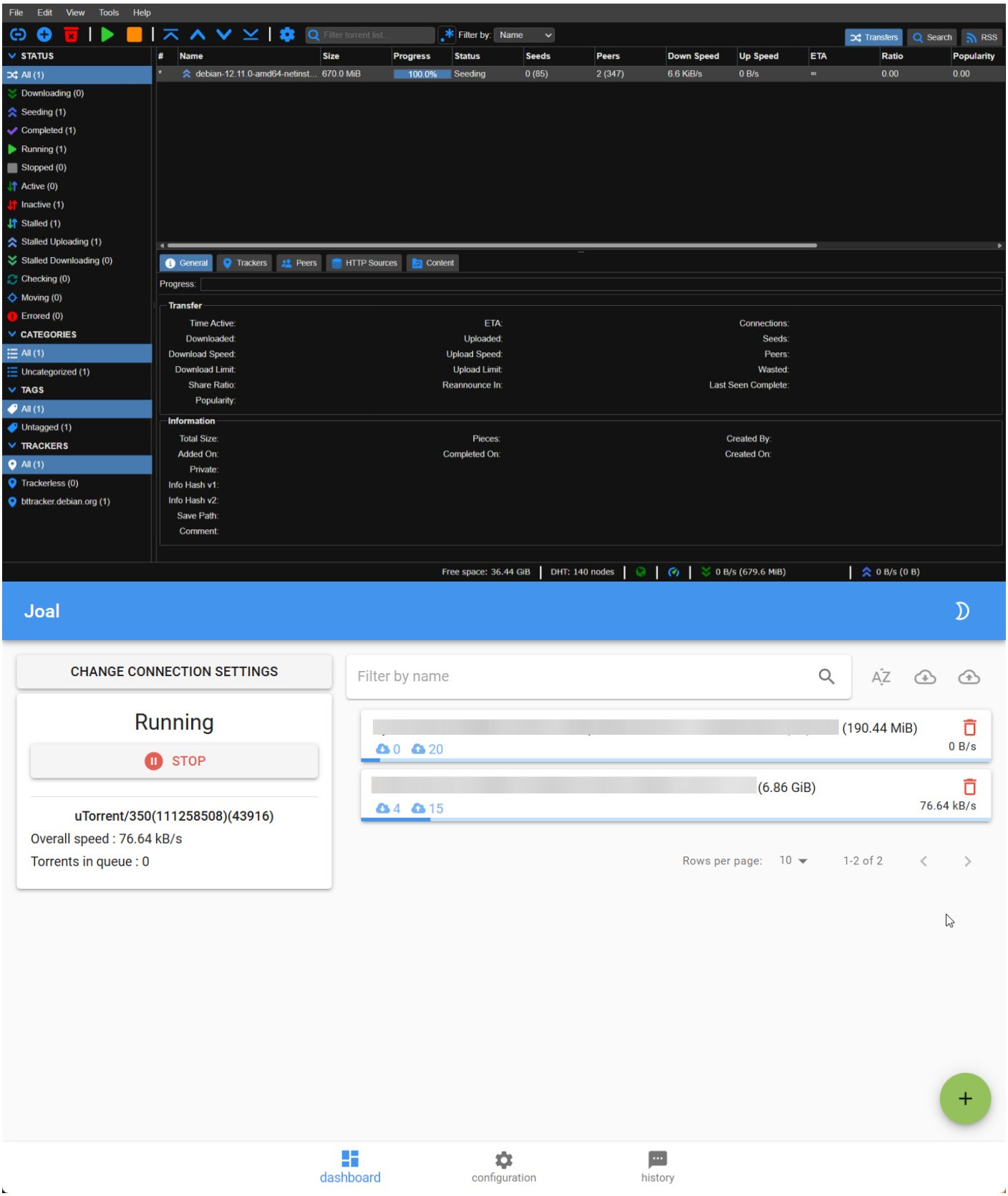
```
http://<ip_serveur>:5800
```

Compose

Téléchargez avec qBittorrent et boostez votre ratio grâce à Joal, en restant derrière un VPN

Ce guide présente une configuration Docker pour télécharger des torrents tout en simulant une activité crédible sur les trackers, le tout en restant derrière un VPN.

> ⚠ Ce wiki n'est pas conçu pour vous aider ou vous encourager à télécharger des contenus illégaux ! Vous devez respecter la législation en vigueur dans votre pays. Je ne saurais être tenu responsable des activités illégales résultant de votre utilisation.



1. Docker Compose pour qBittorrent et Joal

Voici un exemple de fichier `docker-compose.yml` configuré pour :

- **Gluetun** : un conteneur VPN robuste prenant en charge de nombreux fournisseurs (Surfshark, NordVPN, Mullvad, etc.) pour masquer votre adresse IP et protéger votre

anonymat.
- **test_ip_vpn** : un outil léger pour vérifier que le VPN est bien actif et que votre IP publique correspond à celle du VPN.
- **qBittorrent** : le client torrent puissant et facile à utiliser, accessible via une interface web pour gérer vos téléchargements.
- **Joal** : un simulateur d'activité torrent permettant d'envoyer de faux partages, afin de retarder ou d'éviter les restrictions liées au ratio sur certains trackers.

Important :
Adaptez les chemins des volumes en fonction de l'emplacement de vos données sur votre système.

Le conteneur **Gluetun** supporte un grand nombre de fournisseurs VPN, notamment :
AirVPN, Cyberghost, ExpressVPN, FastestVPN, Giganews, HideMyAss, IPVanish, IVPN, Mullvad, NordVPN, Perfect Privacy, Privado, Private Internet Access, PrivateVPN, ProtonVPN, PureVPN, SlickVPN, Surfshark, TorGuard, VPNSecure.me, VPNUnlimited, VyprVPN, WeVPN, Windscribe.

Pour plus d'informations et pour consulter la documentation complète, rendez-vous sur :

https://github.com/qdm12/gluetun

```
services:
  gluetun:
    container_name: gluetun
    image: qmcgaw/gluetun
    restart: unless-stopped
    cap_add:
      - NET_ADMIN
    devices:
      - /dev/net/tun:/dev/net/tun
    ports:
      - 6060:8080                       # Interface web Joal
      - 7070:7070                       # Interface web Qbittorent
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Zurich
      - VPN_SERVICE_PROVIDER=surfshark    # Configs des autres fournisseurs :
https://github.com/qdm12/gluetun-wiki/tree/main/setup/providers
      - VPN_TYPE=wireguard
      - WIREGUARD_PRIVATE_KEY=XXXXXX=     # Remplacez par votre clé privée
      - WIREGUARD_ADDRESSES=10.14.0.2/16
      - SERVER_COUNTRIES=Switzerland
```

```
      - UPDATER_VPN=12h                   # Mets à jour le VPN
      - DNS_ADDRESS=194.242.2.2           # DNS Mullvad
    networks:
      - gluetenbridge

  curl:
    image: curlimages/curl
    container_name: test_ip_vpn
    command: ["curl", "ipinfo.io"]
    restart: no
    network_mode: service:gluetun
    depends_on:
      gluetun:
        condition: service_healthy

  qbittorrent:
    image: lscr.io/linuxserver/qbittorrent:latest
    container_name: qbittorrent
    restart: unless-stopped
    network_mode: service:gluetun
    depends_on:
      gluetun:
        condition: service_healthy
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Zurich
      - WEBUI_PORT=7070
      - TORRENTING_PORT=6881
    volumes:
      - /srv/qbittorrent/appdata:/config  # Assurez-vous que le dossier correspond à votre
configuration
      - /mnt/nas_multimedia:/downloads  # Assurez-vous que le dossier correspond à votre
configuration

#  joal:
#    image: ghcr.io/skylanix/joal:latest
#    container_name: joal
#    restart: unless-stopped
#    network_mode: service:gluetun
```

```
#    depends_on:
#      gluetun:
#        condition: service_healthy
#    environment:
#      - PUID=1000
#      - PGID=1000
#      - TZ=Europe/Zurich
#    volumes:
#      - /srv/joal/data:/data  # Assurez-vous que le dossier correspond à votre configuration
#    command:
#      - "--joal-conf=/data"
#      - "--spring.main.web-environment=true"
#      - "--server.port=8080"
#      - "--joal.ui.path.prefix=joal"
#      - "--joal.ui.secret-token=YourToken"


networks:
  gluetenbridge:
    driver: bridge
```

Vérification de l'IP VPN

Le service `curl` est là pour vérifier l'IP publique utilisée par le VPN. Affichez ses logs après lancement :

```
docker logs test_ip_vpn
```

Vous devriez voir l'adresse IP (celle du VPN). Si elle ne s'affiche pas, redémarrez le conteneur, vous vérifiez à nouveau que vous êtes bien connecté au VPN.

?? Avertissements importants

- **Sécurité du token :**
  Remplacez `YourToken` par un mot de passe fort unique. Ne le divulguez pas publiquement et protégez l'accès à l'interface Joal.
- **Gestion du simulateur Joal :**
  Si votre ratio est déjà satisfaisant, pour éviter les risques de bannissement par les trackers, désactivez Joal temporairement en commentant la ligne `restart: unless-stopped` dans le fichier `docker-compose.yml`. Cela empêchera le conteneur de redémarrer automatiquement (le simulateur démarre un upload dès son lancement).

2. Accéder à l'interface web de qBittorent

L’interface est accessible via :

```
http://<ip_serveur>:7070
```

Le nom d'utilisateur est admin et le mot de passe temporaire est a changé ce trouve dans les logs du conteneur :

```
docker logs qbittorrent
```

3. Accéder à l’interface web de Joal

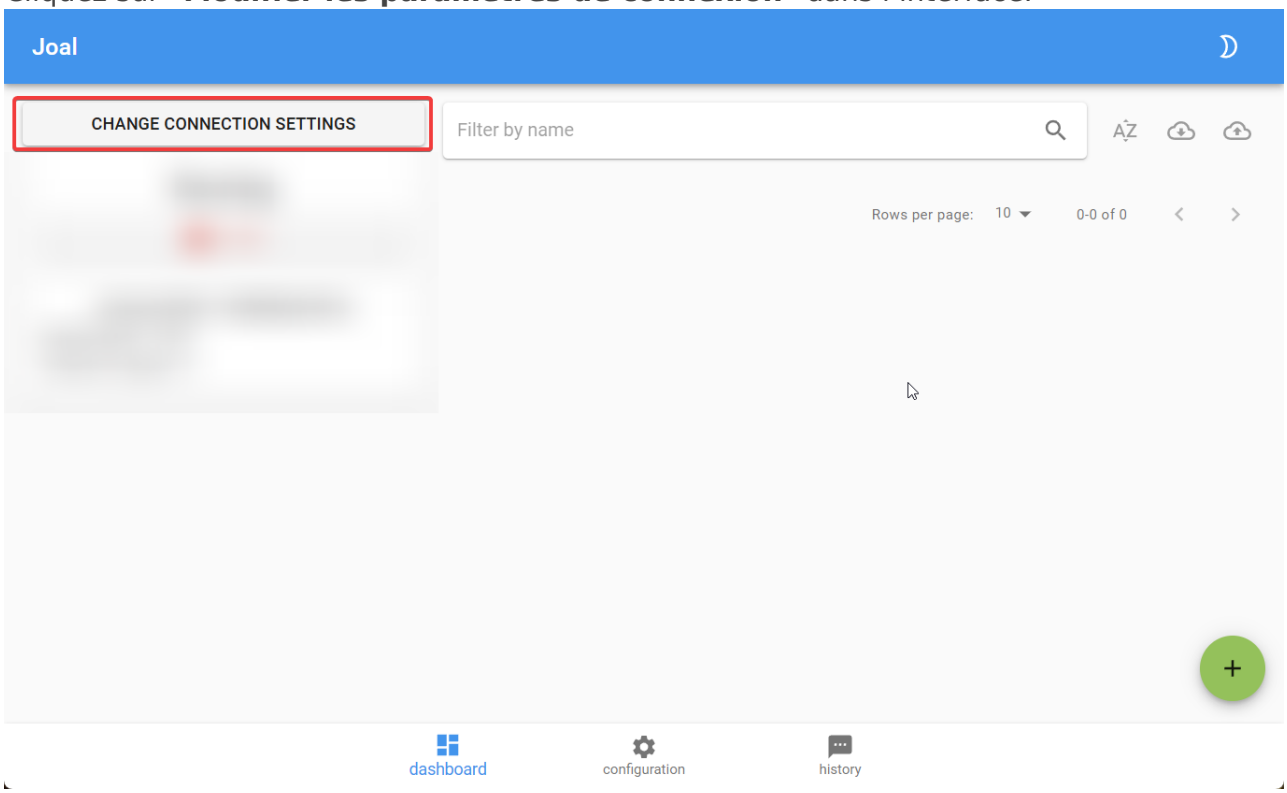
L’interface est accessible via :

```
http://<ip_serveur>:6060/joal/ui
```

Note : Le chemin `/joal/ui` correspond au préfixe défini dans la configuration.

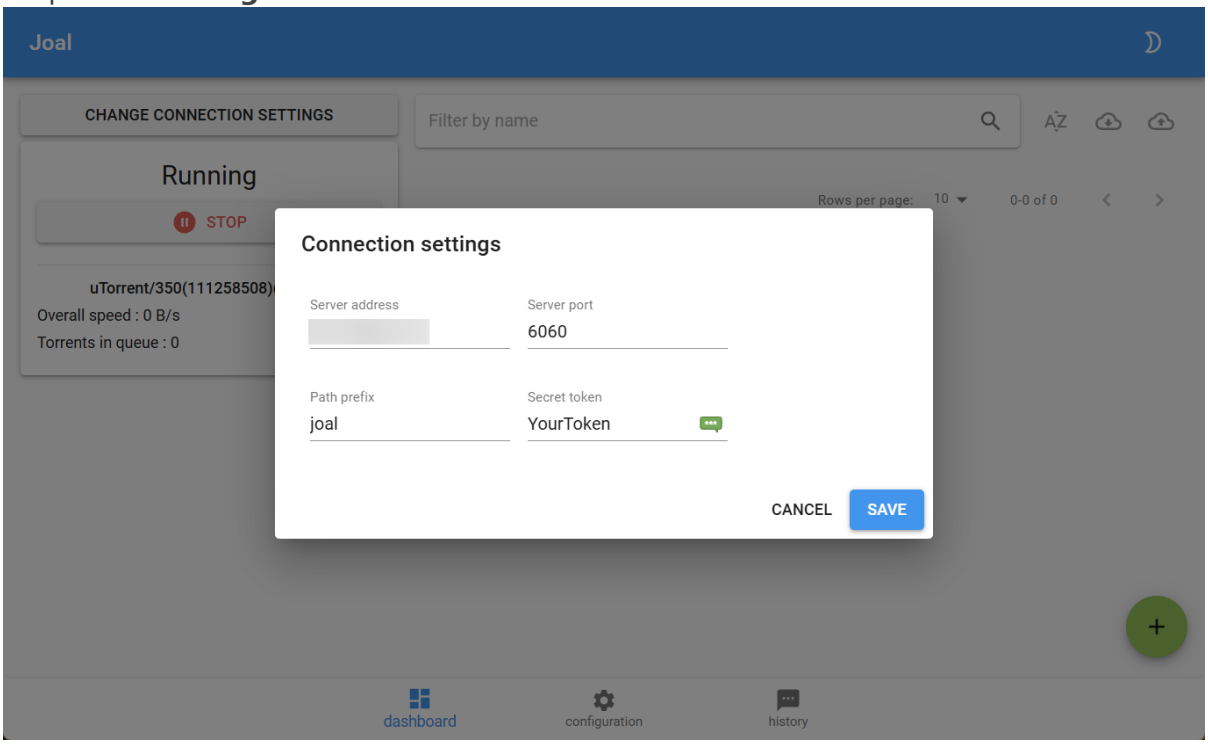
4. Configuration initiale de Joal

1. Cliquez sur **"Modifier les paramètres de connexion"** dans l’interface.



2. Configurez les paramètres :
   - **Adresse du serveur :** l’IP ou le nom de domaine de votre serveur
   - **Port du serveur :** `6060`
   - **Préfixe du chemin :** `joal`
   - **Jeton secret :** le token choisi dans le docker-compose

- Cliquez sur **Enregistrer**



5. Paramètres de configuration Joal

- Définissez les vitesses minimales et maximales de téléchargement simulé.
- Choisissez votre client torrent préféré.
- Limitez le nombre de torrents envoyés simultanément.
- Précisez le ratio requis pour chaque torrent (mettre `-1` pour illimité).

Conseil : Ne poussez pas les vitesses trop haut pour rester discret.

6. Ajout de torrents dans Joal

Trois méthodes possibles :

- Glisser-déposer les fichiers `.torrent` dans le tableau de bord Joal.
- Cliquer sur le bouton "+" et sélectionner un fichier.
- Copier les fichiers `.torrent` directement dans le dossier `/srv/joal/data/torrents`.

7. Conseils d’utilisation

- Sélectionnez de préférence des torrents avec beaucoup de seeds (sources) pour que le partage simulé soit crédible.
- Surveillez régulièrement l’état via l’interface Joal.
- Tenez compte des règles du tracker pour éviter d’être banni.

Si vous avez besoin d’aide supplémentaire ou de précisions, n’hésitez pas à demander !

Remarque : Ce tutoriel s'adresse à un usage modéré et responsable des outils de torrenting. Tricher peut nuire à l’écosystème des trackers et violer leurs conditions d’utilisation.

8. Remerciements

Je remercie Linuxon, Mow, Chomiam et tous les autres d'avoir été les crash-testeurs, de m'avoir rapporté toutes mes erreurs pour améliorer cet article !

Compose

Watchtower

Outil qui surveille automatiquement vos conteneurs Docker et, si nécessaire, peut les mettre à jour pour garantir qu'ils disposent toujours de la dernière version. Il facilite la gestion de vos applications en s'assurant qu'elles restent à jour sans intervention manuelle.

Précautions avant la mise à jour automatique

Je vous recommande vivement de ne pas activer la mise à jour automatique. Il est important de **relire les notes de version** avant de procéder à une mise à jour, afin de suivre les changements et d’éviter tout problème ou incompatibilité qui pourrait casser votre environnement.

Fonctionnement de Watchtower

- Surveille automatiquement vos conteneurs Docker.
- Si une mise à jour est disponible, il peut, selon la configuration, la télécharger et l’appliquer.
- Peut envoyer des notifications pour vous tenir informé des actions effectuées ou en cours.

Note : Il est conseillé de configurer Watchtower en mode **surveillance ONLY** (`WATCHTOWER_MONITOR_ONLY: "true"`) si vous souhaitez d’abord recevoir des notifications sans qu’il procède automatiquement à la mise à jour.

Configuration de Webhook pour notifications

Pour recevoir des notifications dans votre serveur Discord lorsque Watchtower effectue une action (mise à jour, erreur, etc.), vous pouvez utiliser la fonction de webhook de Discord.

Comment créer un webho, sans attendre la mise à jour automatique.ok Discord ?

1. Ouvrez Discord et allez dans le serveur où vous souhaitez recevoir les notifications.
2. Sélectionnez le canal où vous souhaitez envoyer les messages (ex : #watchtower).
3. Cliquez sur la petite roue dentée à côté du nom du canal pour ouvrir ses paramètres.
4. Dans le menu de gauche, cliquez sur **Intégrations** puis sur **Webhooks**.
5. Cliquez sur **Créer un webhook**.
6. Personnalisez le nom du webhook, choisissez une image si vous le souhaitez, puis cliquez sur **Copier le webhook URL**.

7. Conservez cette URL, vous en aurez besoin pour la configuration.

Intégrer le webhook dans votre configuration Watchtower

- Pour recevoir des notifications dans votre serveur Discord, vous devez utiliser la variable d'environnement `WATCHTOWER_NOTIFICATION_URL` dans votre fichier de configuration Docker compose ou dans vos variables d'environnement.
- Remplacez la valeur `"discord://token@id"` par les valeurs de l'URL votre webhook Discord :

  ```
  https://discord.com/api/webhooks/ID/TOKEN
  ```

- Exemple de configuration :

```
environment:
  WATCHTOWER_NOTIFICATION_URL: "discord://M8q5hBYCHqwFTD3R838KIbb-
CIWhGNTPLtHM1wd9S3vlqEIuZGmlNT78Pnur04p46aVq@1416880135902335086"
```

Exemple modifié :

```
services:
  watchtower:                                    # Permet de mettre à jour
automatiquement
    image: containrrr/watchtower:latest
    container_name: watchtower                   # Nom du conteneur
    restart: unless-stopped                      # Redémarre automatiquement sauf si
arrêté manuellement
    environment:
      TZ: Europe/Paris                           # Fuseau horaire
      WATCHTOWER_NOTIFICATIONS_HOSTNAME: "nom"   # Nom d'hôte
#      WATCHTOWER_INCLUDE: "nginx-proxy portainer"     # Liste des conteneurs à surveiller
(séparés par des espaces)
      WATCHTOWER_SCHEDULE: "0 8 * * 0"           # Vérification/Notification dimanches à
8 heures
      WATCHTOWER_MONITOR_ONLY: "true"            # Mode surveillance uniquement : détecte
les mises à jour mais ne les applique pas
      WATCHTOWER_CLEANUP: "true"                 # Supprime automatiquement les anciennes
images après mise à jour
      WATCHTOWER_INCLUDE_RESTARTING: "false"     # Inclut les conteneurs en cours de
redémarrage dans la surveillance
```

```
                                                    # URL de notification Discord (webhook)
      WATCHTOWER_NOTIFICATION_URL: "discord://token@id"
      WATCHTOWER_NOTIFICATION_TEMPLATE: |
        {{range .}}{{.Time.Format "01-02-2006 15:04:05"}} ({{.Level}}):
{{.Message}}{{println}}{{end}}
      WATCHTOWER_NOTIFICATIONS: shoutrrr
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

Remarques importantes :

- Si vous utilisez un autre service de notifications, adaptez l'URL en conséquence en suivant la documentation officielle

Mettre à jour un conteneur avec Watchtower manuellement

Vous pouvez lancer Watchtower une seule fois et à tout moment, avec les commandes ci-dessous.

Mettre à jours un seul conteneur Docker et supprime l'ancienne image :

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock containrrr/watchtower --run-once
--cleanup nom_du_conteneur
```

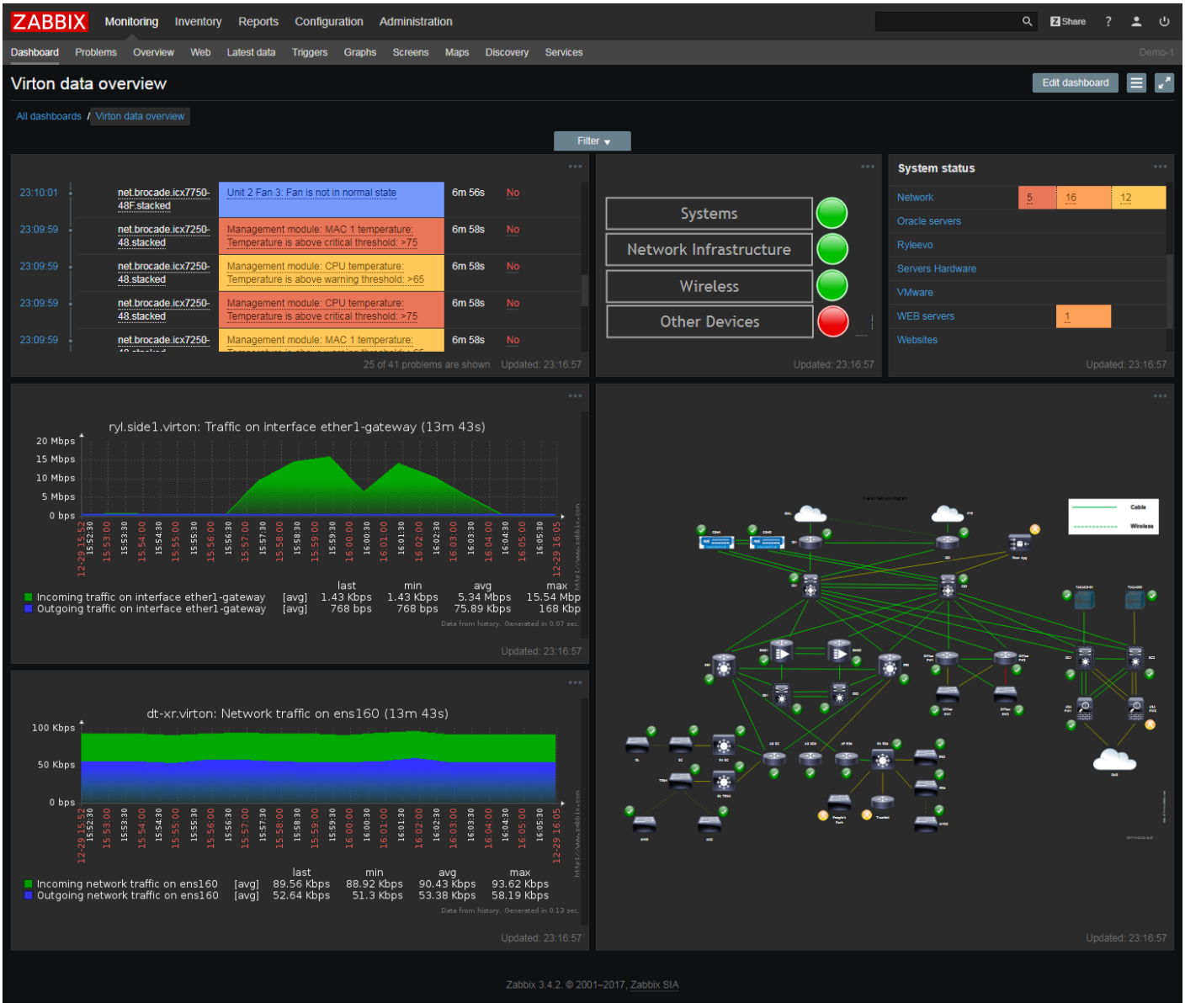
Tout les conteneurs et supprime l'ancienne image :

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock containrrr/watchtower --run-once
--cleanup
```

Compose

Zabbix

Un outil open source de surveillance des infrastructures IT, offrant des alertes et des analyses en temps réel.



> ⚠ **Avertissements importants**
>
> - Remplacez le mot de passe par défaut `Mot_de_passe` dans les **variables** du Docker compose et dans la base de données par un mot de passe **fort et unique**.

> - Protégez l'accès à l'interface Zabbix en limitant son exposition sur Internet (utilisez un VPN, une authentification forte ou un **pare-feu**).
> - Remplacez le mot de passe de l'administrateur **par défaut** de Zabbix.

1. Crée les volumes.

C'est optionnel, si vous ne les utilisez pas, supprimez-les en conséquence dans le compose et faites des montages liés à vos dossiers locaux par exemple `/srv/zabbix/exemple_data`

```
docker volume create postgresql-data
docker volume create zabbix-server-data
docker volume create zabbix-snmptraps-data
docker volume create zabbix-export-data
docker volume create zabbix-web-data
```

2. Docker Compose

```
services:
  postgresql-server:
    image: postgres:latest
    container_name: postgresql-server
    restart: unless-stopped
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
      - postgresql-data:/var/lib/postgresql/data        # À modifier si vous faites des
montages liés à vos dossiers locaux

  zabbix-server:
    image: zabbix/zabbix-server-pgsql:latest
    container_name: zabbix-server
    restart: unless-stopped
    depends_on:
      - postgresql-server
    environment:
      DB_SERVER_HOST: postgresql-server
```

```
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    ports:
      - "10051:10051"                                  # Port d'écoute agent
    volumes:
      - zabbix-server-data:/var/lib/zabbix             # À modifier si vous voulez plutôt
faire des montages liés à vos dossiers locaux
      - zabbix-snmptraps-data:/var/lib/zabbix/snmptraps # À modifier si vous voulez plutôt
faire des montages liés à vos dossiers locaux
      - zabbix-export-data:/var/lib/zabbix/export       # À modifier si vous voulez plutôt
faire des montages liés à vos dossiers locaux

  zabbix-web-nginx-pgsql:
    image: zabbix/zabbix-web-nginx-pgsql:latest
    container_name: zabbix-web
    restart: unless-stopped
    depends_on:
      - postgresql-server
      - zabbix-server
    environment:
      DB_SERVER_HOST: postgresql-server
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
      ZBX_SERVER_HOST: zabbix-server
      PHP_TZ: ${PHP_TZ}
    ports:
      - "${ZABBIX_FRONTEND_PORT}:8080"
    volumes:
      - zabbix-web-data:/usr/share/zabbix               # À modifier si vous voulez plutôt
faire des montages liés à vos dossiers locaux

  zabbix-agent:
    image: zabbix/zabbix-agent:latest
    container_name: zabbix-agent
    restart: unless-stopped
    depends_on:
      - zabbix-server
    environment:
```

```
      ZBX_HOSTNAME: "zabbix-server"
      ZBX_SERVER_HOST: zabbix-server
      ZBX_SERVER_PORT: '10051'                              # Communique avec le port d'écoute
agent du Zabbix server
      ZBX_SERVER_ACTIVE: zabbix-server
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock

volumes:                                                    # À supprimez si vous faites des
montages liés à vos dossiers locaux
  postgresql-data:
      external: true
  zabbix-server-data:
      external: true
  zabbix-snmptraps-data:
      external: true
  zabbix-export-data:
      external: true
  zabbix-web-data:
      external: true
```

3. Environnement variable

```
POSTGRES_USER=zabbix
POSTGRES_PASSWORD=Mot_de_passe                          # Remplacez le mot de passe par un mot
de passe fort
POSTGRES_DB=zabbix
PHP_TZ=Europe/Zurich                                    # Remplacez par votre
ZABBIX_FRONTEND_PORT=17318                              # Optionnel : Remplacez le port de la
page Zabbix
```

4. Accédez à la page d'administration

`ip:17318` et le nom d'utilisateur par défaut est `Admin` et le mot de passe `zabbix`

5. Notification Discord

https://www.youtube.com/embed/cqnHWhDt8Ec

6. Installation de Zabbix Agent sur une base Debian

Pour surveiller d'autres serveurs Debian, vous devez installer l'agent Zabbix sur chaque machine à monitorer.

6.1. Installation via le dépôt officiel Zabbix

```
# Mettre à jour les paquets
sudo apt update

# Installer les dépendances
sudo apt install -y wget curl gnupg2 software-properties-common

# Télécharger et installer le dépôt Zabbix (Debian 13 - Trixie)
wget https://repo.zabbix.com/zabbix/7.0/debian/pool/main/z/zabbix-release/zabbix-release_latest+debian13_all.deb
sudo dpkg -i zabbix-release_latest+debian13_all.deb
sudo apt update

# Installer l'agent Zabbix
sudo apt install -y zabbix-agent2

# Installer les plugins Zabbix agent 2
sudo apt install -y zabbix-agent2-plugin-mongodb zabbix-agent2-plugin-mssql zabbix-agent2-plugin-postgresql
```

> **Note :** Pour Debian 12 (Bookworm), remplacez `debian13` par `debian12` dans l'URL ci-dessus.

Pour les autres distributions : https://www.zabbix.com/download

6.2. Configuration de l'agent

Éditez le fichier de configuration de l'agent :

```
sudo nano /etc/zabbix/zabbix_agent2.conf
```

Modifiez les paramètres suivants :

```
# Adresse IP ou hostname du serveur Zabbix
Server=IP_DU_SERVEUR_ZABBIX

# Adresse IP du serveur Zabbix pour les checks actifs
ServerActive=IP_DU_SERVEUR_ZABBIX

# Nom d'hôte unique pour identifier ce serveur dans Zabbix
Hostname=nom-du-serveur-debian

# Optionnel : Autoriser les commandes à distance (à activer avec prudence)
# AllowKey=system.run[*]
```

Exemple avec l'IP `192.168.1.100` :

```
Server=192.168.1.100
ServerActive=192.168.1.100
Hostname=debian-web-01
```

6.3. Démarrer et activer l'agent

```
# Démarrer l'agent Zabbix
sudo systemctl start zabbix-agent2

# Activer le démarrage automatique
sudo systemctl enable zabbix-agent2

# Vérifier le statut
sudo systemctl status zabbix-agent2
```

6.4. Configurer le pare-feu

Si vous utilisez UFW ou iptables, autorisez le port 10050 :

```
# Avec UFW
sudo ufw allow 10050/tcp

# Avec iptables
sudo iptables -A INPUT -p tcp --dport 10050 -j ACCEPT
sudo netfilter-persistent save
```

6.5. Ajouter l'hôte dans l'interface Zabbix

1. Connectez-vous à l'interface web Zabbix
2. Allez dans **Configuration** → **Hosts**
3. Cliquez sur **Create host**
4. Remplissez les informations :
   - **Hostname** : `debian-web-01` (le même que dans la config)
   - **Groups** : Sélectionnez ou créez un groupe (ex: "Linux servers")
   - **Agent** : Dans l'onglet Interfaces, ajoutez l'IP de votre serveur Debian
   - **Port** : `10050`
5. Dans l'onglet **Templates**, ajoutez le template `Linux by Zabbix agent`
6. Cliquez sur **Add**

6.6. Vérification

Pour vérifier que l'agent communique correctement :

```
# Vérifier les logs
sudo tail -f /var/log/zabbix/zabbix_agent2.log

# Tester la connectivité depuis le serveur Zabbix
zabbix_get -s IP_DU_SERVEUR_DEBIAN -k agent.ping
```

> **Astuce :** Dans l'interface Zabbix, l'icône de disponibilité de l'hôte deviendra verte (ZBX) après quelques minutes si la connexion est établie.

7. Si vous voulez approfondir, Zabbix propose une bonne documentation

Compose

Effacement sécurisé complet du disque

Remplacez le disque `/dev/sdX` par votre choix :

```
services:
  hdparm-secure-erase:
    container_name: hdparm-secure-erase-sdX
    image: ubuntu:latest
    privileged: true
    tty: true
    stdin_open: true
    devices:
      - /dev/sdX:/dev/sdX
    command:
      - /bin/bash
      - -c
      - |
        apt-get update && apt-get install -y hdparm
        echo '=== ATTENTION: Effacement sécurisé de /dev/sdX ==='
        hdparm -I /dev/sdX | grep -i security
        echo 'Définition du mot de passe...'
        hdparm --user-master u --security-set-pass PASSWORD /dev/sdX
        echo "Lancement de l'effacement sécurisé..."
        hdparm --user-master u --security-erase-enhanced PASSWORD /dev/sdX
        echo 'Vérification finale...'
        hdparm -I /dev/sdX | grep -i security
```