Docker

Parce que l'air marin fait aussi du bien

- Installation de Docker (draft)
- Jackett Radarr Sonnar Lidarr Ombi Tautulli & Flaresolverr Traefik dans Docker
- New Page

Installation de Docker (draft)

Installation de Docker

Debian/Ubuntu

- Installation des pré-requis.

```
apt-get update && apt-get upgrade && apt-get install
 apt-transport-https ca-certificates xz-utils iptables aufs-tools git
```

- On ajoute les dépôts.

```
echo "deb https://apt.dockerproject.org/repo debian-jessie main" > /etc/apt/
sources.list.d/docker.list
apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
 58118E89F3A912897C070ADBF76221572C52609D
```

- Puis on installe docker.

```
apt-get update && apt-get install docker-engine
```

ArchLinux/Manjaro

- On installe docker.

```
pacman -Syu && pacman -S docker
```

Activation de Docker

- Il ne reste plus qu'à le lancer.

```
systemctl start docker
systemctl enable docker
```

- On test notre installation avec l'image de test.

```
$ docker run hello-world
```

Source: https://wiki.hutit.fr/capharnaum/docker

Jackett Radarr Sonnar Lidarr Ombi Tautulli & Flaresolverr Traefik dans Docker

Bonjour à tous, je vous propose un petit tutoriel clé en main pour installer la suite Jackett Radarr Sonnar Lidarr Ombi Tautulli & Flaresolverr dans Docker (avec Docker-Compose) et en utilisant Traefik en reverse proxy.

Petite présentation :

La suite que je vous propose aujourd'hui, permet entre autre d'automatiser le télécharger de torrent, soit pour votre Plex (ou autre) ou simplement pour vous !

Petit tour des utilités des différents services :

- **Sonarr** permet de rechercher vos fichiers *.torrent* et d'automatiser le téléchargement de vos séries préférées. Vous ajoutez une série dans l'interface en précisant la qualité et la langue souhaitées et Sonarr recherchera celle-ci via les indexers configurés. Enfin Sonarr ajoutera automatiquement dans votre client torrent la série en téléchargement. Parmi les nombreuses fonctionnalités de Sonarr, celui-ci dispose de tâches automatiques et quotidiennes ajoutant automatiquement un épisode fraîchement sorti. L'interface dispose aussi d'un calendrier répertoriant les sorties des prochains épisodes.
- **Radarr** et **Lidarr** ont un fonctionnement similaire à **Sonarr** mais respectivement pour les films et les fichiers audio.
  Cependant Sonarr et Radarr ne proposent que très peu d'indexers (ou trackers) aujourd'hui.
- **Jackett** permet de combler ce manque et prend en charge plus d'une centaine de trackers. De nombreux trackers Français (YGGtorrent) sont supportés ainsi que des trackers privés et semi-privés nécessitant un compte.
  Jackett fonctionne comme un serveur proxy, lorsque vous effectuez une recherche via Sonarr ou Radarr, celui-ci transforme et transmet la requête au tracker, analyse la réponse puis renvoie les résultats à l'application émettrice. Jackett prend aussi en charge les flux RSS.
- **Ombi** est une interface web qui donne la possibilité à vos utilisateurs Plex de demander du contenu par eux-mêmes ! Ombi peut être lié à Sonarr, Radarr ou encore Headphones pour lancer le téléchargement automatiquement. Une fois le média disponible dans Plex,

l’utilisateur à l’origine de la requête pourra être alors notifié par email, Pushbullet, Pushover ou encore Slack.
- **Tautulli** (anciennement PlexPy), un programme de monitoring codé en python basé sur Headphones et PlexWatchWeb, vous permettra d’obtenir, via son interface web, de nombreuses statistiques sur l’activité des utilisateurs de votre Plex (qui sont les utilisateurs qui s’y connectent ? Que regardent-ils ? Quels sont les films et les séries les plus regardés ?) mais aussi des données sur votre serveur (mémoire utilisée, charge CPU, etc.).
- **Flaresolverr** est un serveur proxy pour contourner la protection Cloudflare. (très utile pour YGGTorrent !)

Avec cette suite de services, vous serez capable d'avoir un Plex presque autonome pour l'ajout de contenu, et aussi permettre à votre famille et amis de pouvoir faire des demandes d'ajouts sans recevoir des messages à 4h du matin pour un film xD

Pour commencer, on va installer les pré-requis :

```
sudo apt install -y curl software-properties-common && apt update
```

Maintenant, installons Docker et Docker-Compose :

```
sudo apt install -y docker.io
```

Vérifions qu'il n'y a pas eu de soucis avec l'installation de Docker en faisant la commande : (Actuellement nous sommes en version Docker version 20.10.7 pour Ubuntu et Docker version 20.10.5+dfsg1 pour Debian 11)

```
docker -v
```

Pour installer Docker-Compose nous allons faire ces commandes :

```
sudo curl -L https://github.com/docker/compose/releases/download/1.27.4/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose - sudo chmod +x /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

et vous allons revérifier que tout est bon en vérifiant la version de docker : (Actuellement nous sommes en version docker-compose 1.27.4)

```
docker-compose -v
```

Maintenant que les pré-requis et Docker sont installés, nous allons rentrer dans le sujet.

On commence par la partie Traefik (on garde le plus facile pour la fin, l'effort puis le réconfort)

On crée le dossier qui va accueillir Traefik :

```
mkdir Traefik
```

Maintenant on va créer notre network traefik

```
docker network create traefik_network
```

Créer le fichier *docker-compose.yml* dans le dossier Traefik qu'on a crée juste avant et coller ceci dedans (il n'y a pas besoin de le modifier) :

```
version: '3.7'
services:
traefik:
image: "traefik:v2.2"
container_name: "traefik"
restart: unless-stopped
networks:
- traefik_network
ports:
- "80:80"
- "443:443"
volumes:
- /var/run/docker.sock:/var/run/docker.sock:ro
- /home/traefik/traefik.toml:/traefik.toml:ro
- /home/traefik/acme.json:/acme.json

networks:
traefik_network:
external: true
```

Une fois fait, créer le fichier *acme.json* qui servira à stocker nos certificats :

```
touch acme.json
```

et créer le fichier *traefik.toml* qui est votre fichier de configuration Traefik et coller ceci : (la seule et unique chose à modifier est l'adresse mail !)

```
[providers.docker]
endpoint = "unix:///var/run/docker.sock"
watch = true
exposedByDefault = false

[entryPoints.web]
address = ":80"
[entryPoints.web.http.redirections.entryPoint]
to = "websecure"
scheme = "https"

[entryPoints.websecure]
address = ":443"

[certificatesResolvers.leresolver.acme]
email = "monsupermail@ataxya.genial"
storage = "acme.json"
[certificatesResolvers.leresolver.acme.httpChallenge]
entryPoint = "web"
```

et enfin lancer le docker :

```
docker-compose up -d
```

Maintenant passons au plus simple, le docker-compose pour la suite d'outils :
Sortez du dossier de Traefik et créer un dossier du nom de **Media** :

```
mkdir Media && cd Media
```

Créer le fichier *docker-compose.yml* et coller ceci dedans : (normalement rien n'est à modifier) :

```
version: '3.7'
services:
jackett:
image: linuxserver/jackett:latest
container_name: jackett
restart: unless-stopped
networks:
- ${NETWORK}
environment:
```

```
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/jackett/config:/config
- /etc/localtime:/etc/localtime:ro
labels:
- "traefik.enable=true"
- "traefik.docker.network=${NETWORK}"
- "traefik.http.routers.jackett.entrypoints=web,websecure"
- "traefik.http.routers.jackett.rule=Host(`jackett.${BASE_HOST}`)"
- "traefik.http.services.jackett.loadbalancer.server.port=9117"
- "traefik.http.routers.jackett.tls=true"
- "traefik.http.routers.jackett.tls.certresolver=leresolver"

sonarr:
image: linuxserver/sonarr:latest
container_name: sonarr
restart: unless-stopped
networks:
- ${NETWORK}
environment:
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/sonarr/config:/config
- ${PATH_MEDIA}:/data
labels:
- "traefik.enable=true"
- "traefik.docker.network=${NETWORK}"
- "traefik.http.routers.sonarr.entrypoints=web,websecure"
- "traefik.http.routers.sonarr.rule=Host(`sonarr.${BASE_HOST}`)"
- "traefik.http.services.sonarr.loadbalancer.server.port=8989"
- "traefik.http.routers.sonarr.tls=true"
- "traefik.http.routers.sonarr.tls.certresolver=leresolver"

radarr:
image: linuxserver/radarr:latest
container_name: radarr
```

```
restart: unless-stopped
networks:
- ${NETWORK}
environment:
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/radarr/config:/config
- ${PATH_MEDIA}:/data
labels:
- "traefik.enable=true"
- "traefik.docker.network=${NETWORK}"
- "traefik.http.routers.radarr.entrypoints=web,websecure"
- "traefik.http.routers.radarr.rule=Host(`radarr.${BASE_HOST}`)"
- "traefik.http.services.radarr.loadbalancer.server.port=7878"
- "traefik.http.routers.radarr.tls=true"
- "traefik.http.routers.radarr.tls.certresolver=leresolver"

lidarr:
image: linuxserver/lidarr:latest
container_name: lidarr
restart: unless-stopped
networks:
- ${NETWORK}
environment:
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/lidarr/config:/config
- ${PATH_MEDIA}:/data
labels:
- "traefik.enable=true"
- "traefik.docker.network=${NETWORK}"
- "traefik.http.routers.lidarr.entrypoints=web,websecure"
- "traefik.http.routers.lidarr.rule=Host(`lidarr.${BASE_HOST}`)"
- "traefik.http.services.lidarr.loadbalancer.server.port=8686"
- "traefik.http.routers.lidarr.tls=true"
- "traefik.http.routers.lidarr.tls.certresolver=leresolver"
```

```
ombi:
image: linuxserver/ombi:latest
container_name: ombi
restart: unless-stopped
networks:
- ${NETWORK}
environment:
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/config:/config
- /etc/localtime:/etc/localtime:ro
labels:
- "traefik.enable=true"
- "traefik.docker.network=${NETWORK}"
- "traefik.http.routers.ombi.entrypoints=web,websecure"
- "traefik.http.routers.ombi.rule=Host(`ombi.${BASE_HOST}`)"
- "traefik.http.services.ombi.loadbalancer.server.port=3579"
- "traefik.http.routers.ombi.tls=true"
- "traefik.http.routers.ombi.tls.certresolver=leresolver"

tautulli:
image: linuxserver/tautulli:latest
container_name: tautulli
restart: unless-stopped
networks:
- ${NETWORK}
environment:
- PUID=${PUID}
- PGID=${PGID}
- TZ=Europe/Paris
volumes:
- /home/media/tautulli/config:/config
labels:
- "traefik.enable=true"
- "traefik.docker.network=traefik_network"
- "traefik.http.routers.tautulli.entrypoints=web,websecure"
- "traefik.http.routers.tautulli.rule=Host(`tautulli.${BASE_HOST}`)"
```

```
- "traefik.http.services.tautulli.loadbalancer.server.port=8181"
- "traefik.http.routers.tautulli.tls=true"
- "traefik.http.routers.tautulli.tls.certresolver=leresolver"

flaresolverr:
# DockerHub mirror flaresolverr/flaresolverr:latest
image: ghcr.io/flaresolverr/flaresolverr:latest
container_name: flaresolverr
environment:
- LOG_LEVEL=info
- LOG_HTML=false
- CAPTCHA_SOLVER=none
ports:
- "8191:8191"
restart: unless-stopped
networks:
- traefik_network

networks:
traefik_network:
external: true
```

Maintenant, lancer le docker :

```
docker-compose up -d
```

Une fois fait, aller dans le dossier .env qui a été crée dans le dossier Media et rajouter ceci :

```
PUID=1001
PGID=1001
PATH_MEDIA=/data
NETWORK=traefik_network
BASE_HOST=
```

ATTENTION ! Renseigner bien que la base du nom de domaine sans le www ! exemple **ataxya.cool**, et laisser le NETWORK comme il est.

Maintenant vous pouvez relancer votre docker avec :

```
docker-compose up -d
```

BRAVO ! Vous pouvez vous rendre sur vos différents sites aux adresses suivantes :

- Jackett : **http://jackett.mondomaine.com**
- Sonarr : **http://sonarr.mondomaine.com**
- Radarr : **http://radarr.mondomaine.com**
- Lidarr : **http://lidarr.mondomaine.com**
- Ombi : **http://lidarr.mondomaine.com**
- Tautulli : **http://tautulli.mondomaine.com**

PETITE MISE EN GARDE, car ça vous évitera bien des prises de têtes ensuite (je vous offre le tuyaux car sinon vous allez vous arracher les cheveux)

Pour vos différents sites, si vous devez les liés entre eux à certains moment, l'adresse URL à mettre au début est celle ci :

http://nomDuService:PORT

Pour Flaresolverr à mettre dans Jackett : **http://flaresolverr:8191**

Pour Sonarr / Radarr et Lidarr dans l'indexer, l'adresse est la suivante : (Exemple avec YGGTorrent)

http://jackett:9117/api/v2.0/indexers/yggcookie/results/torznab/

Car si vous prenez juste l'URL torznab dans Jackett il va vous donner l'IP du serveur et c'est pas bon, vous aurez une belle erreur.
La vous avez pas de raison de galérer, mais au besoin, contactez moi sur le Discord d'Ataxya !

New Page