

Altair et ses services

- [OTP](#)
- [Peertube](#)
- [qBittorrent](#)
- [PHP et caddy](#)
- [Nextcloud](#)
- [jellyfin](#)
- [Lamalle](#)
- [libre office online](#)

OTP

Activer l'otp

```
google-authenticator -s ~/.ssh/google_authenticator
```

Config

```
/etc/pam.d/sshd
```

```
auth    required    pam_google_authenticator.so secret=/home/${USER}/.ssh/google_authenticator nullok
```

```
/etc/ssh/sshd_config.d/10-custom.conf
```

```
PasswordAuthentication no
ChallengeResponseAuthentication yes
UsePAM yes
```

Peertube

- `/data/peertube-data` : stockage persistant de peertube
- `/etc/peertube` : dossier contenant les fichiers de configuration
- `/etc/peertube/peertube.nginx.conf` : configuration nginx

Base de données

Installation de postgres :

```
dnf install postgresql-server postgresql-contrib
postgresql-setup --initdb --unit postgresql
systemctl enable postgresql
sudo systemctl start postgresql
```

Création de la base de données :

```
# création de l'utilisateur
sudo -u postgres createuser -P --interactive
# création de la base de donnée
sudo -u postgres createdb -O peertube peertube_prod
```

Autoriser les containers à accéder à la base de données en ajoutant cette ligne à la fin du fichier

`~postgres/data/pg_hba.conf` :

```
# Containers
host    all             all             10.88.0.0/24    md5
```

Pare-feu

Vérifier le firewall pour autorisé le trafic vers / depuis le réseau de podman, pour [nftables](#) ajouter ces lignes à la tables `inet filter` :

```
iifname cni-podman0 accept
oifname cni-podman0 accept
```

Peertube

Lancement de l'instance :

```
# Créer un pod pour Nginx et Peertube
podman pod create --name peertube --restart=unless-stopped -p 127.0.0.1:9000:9000 -p 0.0.0.0:1935:1935 -p 127.0.0.1:9080:9080

# Créer le conteneur Nginx
podman create --name peertube-nginx-1.25.3-1 --pod peertube \
--restart=unless-stopped \
-v /etc/peertube/peertube.nginx.conf:/etc/nginx/conf.d/default.conf:Z \
-v /etc/peertube:/config:Z \
-v /data/peertube-data/storage:/var/www/peertube/storage:Z \
nginx:1.25.3
# Démarrer Nginx
podman start peertube-nginx-1.25.3-1

# Créer le conteneur Peertube
podman create --name peertube-5.0.1-1 --pod peertube \
--restart=unless-stopped \
-v /data/peertube-data:/var/lib/peertube:Z \
-v /etc/peertube:/config:Z \
chocobozzz/peertube:v5.0.1-bullseye
# Démarrer Peertube
podman start peertube-5.0.1-1
```

Configuration Caddy

```
tube.fede.re {
    #respond "Ongoing maintenance, we'll be back soon :)" 503
    reverse_proxy {
        to localhost:9080
    }
}
```

Mise à jour de la configuration Nginx

La configuration à copier est à partir du commentaire `# Application` jusqu'à la fin du bloc. Après copie, il faut supprimer les blocs faisant référence au dossier `peertube-latest/client`.

En cas de problèmes

Au redémarrage d'Altair peertube risque de ne pas redémarrer correctement, si ce n'est pas up vérifier :

- que redis écoute bien sur 10.88.0.1 (`ls -l -i :6379`) sinon le redémarrer via `systemctl restart redis`
- que postgres écoute bien sur 10.88.0.1 (`ls -l -i :5432`) sinon le redémarrer via `systemctl restart postgres`
- Redémarrer le pod: `podman pod start peertube`

Troubleshooting 4.0.0

Python3 n'est pas installé dans le conteneur. Il faut installer python3 et symlink l'exécutable, depuis le conteneur:

- `apt install -y python3`
- `ln -s /usr/bin/python3 /usr/bin/local/python`

qBittorrent

La configuration de qBittorrent est un peu particulière pour éviter de sortir directement avec les IPs d'Altaïr. Un point de sortie (ici Alshain) est utilisé, et tout le trafic de qBittorrent est routé par ce point de sortie. Un tunnel Wireguard assure la liaison entre Altaïr et Alshain. L'IPv6 est routée directement alors que l'IPv4 est NATé (en deux fois, sur Altaïr et sur Alshain pour le moment).

Configuration réseaux :

Serveur	Interace	IP/Masque	Description
Alshain	int-ppsfleet	10.114.20.1/24	Réseau privée interne entre serveur PPSFleet.
Alshain	int-ppsfleet	2001:bc8:24d8:114:20::1/80	Équivalent IPv6 du réseau précédant. Le découpage du réseau est fait par bloc de 16 bits de manière à ce que chaque client aie un /96.
Altaïr	int-ppsfleet	10.114.20.10/24	IPv4 du client VPN sur Altaïr
Altaïr	int-ppsfleet	2001:bc8:24d8:114:20:10:0:1/112	IPv6 du client VPN sur Altaïr. Un sous découpage en /112 par interace dans ce /96 sur Altaïr.
Altaïr	net-external	10.89.0.1/24	Réseau IPv4 des conteneurs sortant via le VPN. Ce réseau est NATé sur Altaïr de manière à ce que le point de sortie ne voit que l'IP du client VPN.
Altaïr	net-external	2001:bc8:24d8:114:20:10:1:0/112	Réseau IPv6 des conteneurs sortant via le VPN.

Configuration du point VPN pour le point de sortie

1. Création les clés pour l'authentification et le chiffrement.

```
# Altaïr
bash -c '(umask 0077; wg genkey > altair.key)'
wg pubkey < altair.key > altair.pub

# Alshain
bash -c '(umask 0077; wg genkey > alshain.key)'
wg pubkey < alshain.key > alshain.pub

# Secret partagé
wg genpsk > altair-alshain.psk
```

2. Configurer le réseau sur Alshain en éditant le fichier `/etc/NetworkManager/system-connections/int-ppsfleet.nmconnection`

```
[connection]
id=int-ppsfleet
type=wireguard
interface-name=int-ppsfleet

[wireguard]
listen-port=51756
private-key=$ALSHAIN_PRIVATE_KEY

[wireguard-peer.$ALTAIR_PUBLIC_KEY]
preshared-key=$ALTAIR_ALSHAIN_SHARED_SECRET
preshared-key-flags=0
allowed-ips=10.114.20.10/32;2001:bc8:24d8:114:20:10::/96;

[ipv4]
address1=10.114.20.1/24
method=manual

[ipv6]
address1=2001:bc8:24d8:114:20::1/80
addr-gen-mode=stable-privacy
method=manual
```

3. Redémarrer NetworkManager sur Alshain

```
sudo systemctl restart NetworkManager
```

Configuration du réseau sur Altaïr

1. Créer la table de routage `external` avec comme ID `200` en ajoutant la ligne suivante au fichier `/etc/iproute2/rt_tables`.

```
200 external
```

2. Configurer le client VPN sur Altaïr en utilisant les clés générées précédemment. Créer le fichier `/etc/NetworkManager/system-connections/int-ppsfleet.nmconnection` avec le contenu suivant.

```
[connection]
id=int-ppsfleet
uuid=fe155cee-4941-3f9b-a441-a7fd21d2412a
type=wireguard
interface-name=int-ppsfleet

[wireguard]
peer-routes=false
private-key=$ALTAIR_PRIVATE_KEY

[wireguard-peer.$ALSHAIN_PUBLIC_KEY]
endpoint=[2001:bc8:24d8::]:51756
preshared-key=$ALTAIR_ALSHAIN_SHARED_SECRET
preshared-key-flags=0
allowed-ips=0.0.0.0/0::/0;

[ipv4]
address1=10.114.20.10/24
method=manual
route-table=200
gateway=10.114.20.1
routing-rule1=priority 5 oif int-ppsfleet table 200

[ipv6]
addr-gen-mode=stable-privacy
address1=2001:bc8:24d8:114:20:10::1/112
method=manual
route-table=200
gateway=2001:bc8:24d8:114:20::1
```



```
routing-rule1=priority 5 oif int-ppsfleet table 200
```

3. Configurer l'interface de bridge pour les conteneurs en créant le fichier

`/etc/NetworkManager/system-connections/net-external.nmconnection` avec le contenu suivant.

```
[connection]
id=net-external
uuid=d72a0e46-85bc-4b4c-bb64-fb904c463894
type=bridge
autoconnect=true
interface-name=net-external

[ethernet]

[bridge]
stp=false

[ipv4]
address1=10.89.0.1/24
method=manual
route-table=200
routing-rule1=priority 5 from 10.89.0.0/24 table 200

[ipv6]
addr-gen-mode=default
address1=2001:bc8:24d8:114:20:10:1:1/112
method=manual
route-table=200
routing-rule1=priority 5 from 2001:bc8:24d8:114:20:10:1:0/112 table 200
```

4. Redémarrer NetworkManager.

```
sudo systemctl restart NetworkManager
```

5. Configurer le firewall pour éviter d'avoir du NAT sur l'IPv6 en ajoutant les lignes suivante au fichier `/etc/sysconfig/nftables.conf`

```
table ip6 nat {
  chain POSTROUTING {
    type nat hook postrouting priority srcnat; policy accept;
    ip6 saddr 2001:bc8:24d8:114:20:10:1:0/112 return
  }
}
```

```
#ifname tun0 masquerade
}
}
```

6. Puis redémarrer nftables.

```
sudo systemctl restart nftables
```

Créer le conteneur qBittorrent sur Altair

1. Créer le réseau Podman

```
nmcli c down net-external
podman network create --subnet 10.89.0.0/24 --subnet 2001:bc8:24d8:114:20:10:1:0/112 --interface-
name=net-external --ignore net-external
nmcli c up net-external
```

2. Démarrer le conteneur qBittorrent une première fois

```
sudo podman run -d --name=qbittorrent \
-e PUID=985 -e PGID=985 -e TZ=Europe/Paris -e WEBUI_PORT=8080 \
-p '[::1]:8080:8080' -p 127.0.0.1:8080:8080 -p 6881:6881 -p 6881:6881/udp \
-v /var/opt/qbittorrent/config:/config:Z -v /data/downloads:/downloads:Z \
--restart always \
--network net-external \
lscr.io/linuxserver/qbittorrent:latest
```

3. Créer une policy SELinux custom (voir [page sur le SELinux du wiki](#)), et re-run le conteneur avec la policy

```
sudo podman run --security-opt label=type:qbittorrent.process \
-d --name=qbittorrent \
-e PUID=985 -e PGID=985 -e TZ=Europe/Paris -e WEBUI_PORT=8080 \
-p '[::1]:8080:8080' -p 127.0.0.1:8080:8080 -p 6881:6881 -p 6881:6881/udp \
-v /var/opt/qbittorrent/config:/config -v /data/downloads:/data/downloads \
--restart always \
--network net-external \
lscr.io/linuxserver/qbittorrent:latest
```

4. Créer le service associé et l'activer

```
sudo podman generate systemd --new --name qbittorrent | sudo tee  
/etc/systemd/system/qbittorrent.service  
systemctl daemon-reload  
systemctl enable qbittorrent
```

PHP et caddy

INSTALL

On a besoin de php-fpm et caddy

Dans le fichier de conf `/etc/php-fpm.d/www.conf`, verifier que

```
listen.allowed_ips = apache,nginx,caddy
```

et

```
user = caddy
; RPM: Keep a group allowed to write in log dir.
group = caddy
```

La conf caddy

Ici on prend comme exemple un nom de domaine "test.ppsfleet.navy"

- creer le fichier `/etc/caddy/Caddyfile.d/test.ppsfleet.navy.conf` avec

```
test.ppsfleet.navy {

    # Set this path to your site's directory.
    root * /srv/www/test.ppsfleet.navy

    php_fastcgi unix//run/php-fpm/www.sock

    # Another common task is to set up a reverse proxy:
    # reverse_proxy localhost:8080

    # Or serve a PHP site through php-fpm:
    # php_fastcgi localhost:9000
```

```
# Refer to the directive documentation for more options.
```

```
# https://caddyserver.com/docs/caddyfile/directives
```

```
}
```

- Autoriser php a executer des scripts dans le dossier

```
chown -R caddy:caddy /srv/www/test.ppsfleet.navy/
```

```
semanage fcontext -a -t httpd_sys_script_exec_t '/srv/www/test.ppsfleet.navy(/.*)?'
```

```
restorecon -Rv /srv/www/test.ppsfleet.navy/
```

Nextcloud

Des notes en vrac

```
php-mysqlnd php-pdo php-pecl-apcu php-xml php-process php-pecl-zip php-gd php-mbstring php-redis php-sodium
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/nextcloud-(.*)/config/(.*)?'
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/nextcloud-(.*)/apps/(.*)?'
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/nextcloud-(.*)/core/(.*)?'
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/.rnd'
```

```
semanage fcontext -a -t var_log_t '/var/log/caddy/(.*)?'
```

```
/srv/nextcloud/php-wsdlcache
```

```
/srv/nextcloud/php-opcache
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/php-session/(.*)?'
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/php-wsdlcache/(.*)?'
```

```
semanage fcontext -a -t httpd_sys_rw_content_t '/srv/nextcloud/php-opcache/(.*)?'
```

jellyfin

Podman way

```
podman run \
    --detach \
    --label "io.containers.autoupdate=registry" \
    --name jellyfin-prod \
    --publish 127.0.0.1:8096:8096/tcp \
    --rm \
    --user $(id -u):$(id -g) \
    --userns keep-id \
    --volume /srv/jellyfin/cache:/cache:Z \
    --volume /srv/jellyfin/config:/config:Z \
    --volume /data/media/video/film:/media/film \
    --volume /data/media/video/series:/media/series \
    --volume /data/media/video/animes:/media/animes \
    --security-opt label=disable \
    docker.io/jellyfin/jellyfin:latest
```

Lamalle

Répertoire de données

Création du répertoire chiffré :

```
truncate -s 3T /data/nextcloud-lamalle.luks
chown nextcloud:nextcloud nextcloud-lamalle.luks
sudo cryptsetup luksFormat nextcloud-lamalle.luks
sudo cryptsetup open nextcloud-lamalle.luks nextcloud-lamalle
sudo mkfs.ext4 -m0 -Lencrypted /dev/mapper/nextcloud-lamalle
```

Monter le répertoire :

```
sudo cryptsetup open /data/nextcloud-lamalle.luks nextcloud-lamalle
sudo mount /dev/mapper/nextcloud-lamalle /mnt/nextcloud-lamalle
```

Démonter le répertoire :

```
sudo mount /mnt/nextcloud-lamalle
sudo cryptsetup close nextcloud-lamalle
```

Source : <https://askubuntu.com/questions/1338610/how-can-i-make-an-encrypted-file-directory-ubuntu-20-04-lts/1338741#1338741>

Docker

```
podman pod create --name lamalle --restart=unless-stopped -p 127.0.0.1:1312:80
```

```
podman create --name lamalle-mariadb-10.6 --pod lamalle \
--restart=unless-stopped \
-v /mnt/nextcloud-lamalle/home/db:/var/lib/mysql:Z \
-e MYSQL_USER=nextcloud \
-e MYSQL_DATABASE=nextcloud \
```



```
-e MYSQL_PASSWORD= \  
-e MYSQL_ROOT_PASSWORD= \  
mariadb:10.6
```

```
podman create --name lamalle-nextcloud-28.0.5 --pod lamalle \  
--restart=unless-stopped \  
-v /mnt/nextcloud-lamalle/home/nextcloud:/var/www/html:Z \  
-e MYSQL_USER=nextcloud \  
-e MYSQL_DATABASE=nextcloud \  
-e MYSQL_PASSWORD= \  
-e MYSQL_HOST=127.0.0.1 \  
nextcloud:28.0.5
```

libre office online

Start

```
systemctl start coolwsd.service
```

Nouvelles polices

Creer les fichiers dans `/usr/share/fonts`

Run:

```
fc-cache  
coolconfig update-system-template
```