Processus de déploiement

Étape 1 — Générer les clés

Sur chaque serveur, générer une paire de clés :

```
# Sur OVH
wg genkey | tee /etc/wireguard/privatekey-ovh | wg pubkey > /etc/wireguard/publickey-ovh

# Sur Marseille
wg genkey | tee /etc/wireguard/privatekey-mrs | wg pubkey > /etc/wireguard/publickey-mrs

# Sur Albi
wg genkey | tee /etc/wireguard/privatekey-albi | wg pubkey > /etc/wireguard/publickey-albi

# Pour chaque client
wg genkey | tee privatekey-client | wg pubkey > publickey-client
```

Étape 2 — Remplacer les placeholders

Dans tous les fichiers .conf, remplacer :

- `<CLE_PRIVEE_OVH>` → contenu de privatekey-ovh
- `<CLE_PUBLIQUE_OVH>` → contenu de publickey-ovh
- `<CLE_PRIVEE_MRS>` → contenu de privatekey-mrs
- `<CLE_PUBLIQUE_MRS>` → contenu de publickey-mrs
- `<CLE_PRIVEE_ALBI>` → contenu de privatekey-albi
- `<CLE_PUBLIQUE_ALBI>` → contenu de publickey-albi
- `<IP_PUBLIQUE_OVH>` → IP publique du VPS OVH
- `<IP_PUBLIQUE_MRS>` → IP publique de la box Marseille
- `<IP_PUBLIQUE_ALBI>` → IP publique de la box Albi
- (idem pour les clés clients)

Étape 3 — Configuration des box internet

Box Marseille & Box Albi

Effectuer dans l'interface admin de chaque box :

Port forwarding :

```
Protocole : UDP
Port externe : 51820
IP destination : 192.168.X.100 (IP locale du serveur)
Port destination : 51820
```

Étape 4 — Déployer dans l'ordre

```
# 1. Démarrer le hub OVH en premier
ssh ovh "cd /opt/wireguard && docker compose up -d"

# 2. Démarrer Marseille
ssh marseille "cd /opt/wireguard && docker compose up -d"

# 3. Démarrer Albi
ssh albi "cd /opt/wireguard && docker compose up -d"

# Une fois wg0 créé, appliquer le pare-feu :
sudo bash ovh-ufw-setup.sh         # sur OVH
sudo bash marseille-ufw-setup.sh   # sur Marseille
sudo bash albi-ufw-setup.sh        # sur Albi
```

Pourquoi le conteneur WireGuard a besoin de host ?

Le point dur, c'est le **routage et le NAT au niveau de l'hôte**. Tes exit-nodes et l'accès LAN reposent sur le fait que le trafic traverse le serveur entre `wg0` et l'interface physique (`enp4s0`, etc.), avec masquerading. Pour que ça fonctionne, `wg0` doit vivre dans le namespace réseau de l'hôte et UFW doit pouvoir filtrer ce trafic au niveau hôte.

Si tu mets WireGuard dans un réseau Docker isolé, `wg0` est créé dans le namespace du conteneur. Tu peux router le mesh entre conteneurs, mais dès que tu veux :

- faire sortir un client sur internet via l'IP publique du serveur (exit node),
- ou atteindre le LAN physique `192.168.x.0/24`,

tu dois faire ressortir le trafic du namespace conteneur vers l'hôte puis vers l'interface physique. C'est faisable mais ça demande des `iptables` supplémentaires et une gymnastique de routage qui

annule l'intérêt de l'isolation. **Pour le rôle de passerelle VPN, host est le choix pragmatique.**

Étape 5 — Vérification de sécurité

Note Docker (importante pour Home Assistant)

Après `ufw enable`, si les conteneurs (Home Assistant, Frigate…) perdent l'accès réseau, redémarrer Docker **une seule fois** pour qu'il replace ses règles FORWARD au-dessus de celles d'UFW :

```
sudo systemctl restart docker
```

À faire à un moment calme (cela redémarre HA et Frigate). En pratique HA en host n'a pas besoin de la chaîne FORWARD, donc ce redémarrage est rarement nécessaire — mais c'est le réflexe si un conteneur perd le réseau.

Vérifier les règles Par-feu

```
ufw status verbose
ufw route status                        # liste les règles FORWARD VPN
iptables -t nat -L POSTROUTING -n | grep MASQUERADE
```

Vérifier que Portainer n'est PAS exposé sur l'IP publique

```
# Depuis internet (doit échouer — connexion refusée ou timeout)
curl -v --max-time 5 http://<IP_PUBLIQUE_OVH>:9000

# Depuis le VPN avec private-only actif (doit fonctionner)
curl -v http://10.0.0.1:9000
```

Tester le cloisonnement LAN

```
# Avec private-only.conf actif, tenter d'atteindre le LAN de Marseille
ping 192.168.1.1    # doit échouer (pas de route + iptables DROP)
ping 192.168.2.1    # doit échouer

# Avec mrs-exitnode.conf actif
ping 192.168.1.1    # doit répondre (autorisé)
ping 192.168.2.1    # doit échouer (bloqué)

# Avec albi-exitnode.conf actif
ping 192.168.1.1    # doit échouer
```

```
ping 192.168.2.1    # doit répondre (autorisé)
```

Vérifier l'état du mesh

```
# Sur OVH — doit lister MRS et Albi avec handshake récent
docker exec wireguard-ovh wg show

# Exemple de sortie attendue :
# peer: <CLE_PUBLIQUE_MRS>
#   endpoint: <IP_MRS>:51820
#   allowed ips: 10.0.0.2/32
#   latest handshake: X seconds ago
#   transfer: X MiB received, X MiB sent
```

Vérifier les sauvegardes rsync (tunnel direct P2P)

```
# Depuis Marseille vers Albi — ne doit PAS faire monter les compteurs OVH
docker exec wireguard-mrs rsync -avz /data/ user@10.0.0.3:/backup/

# Pendant le rsync, vérifier sur OVH que les bytes N'augmentent PAS :
watch -n1 "docker exec wireguard-ovh wg show | grep -A4 'marseille\|albi'"
```

Étape 6 — Configurer Portainer UI

1. Ouvrir http://10.0.0.1:9000 avec le profil private-only.conf actif
2. Aller dans Environments > Add environment > Agent
3. Ajouter Marseille : URL = `10.0.0.2:9001`, nom = "Marseille"
4. Ajouter Albi : URL = `10.0.0.3:9001`, nom = "Albi"

Revision #4
Created 2026-06-05 09:26:47 UTC by gpatruno
Updated 2026-06-05 09:32:03 UTC by gpatruno