Auto-hébergeur OpenSign™ avec Docker sur Linux/MacOS/Windows

Lien :

- https://docs.docker.com/engine/install/debian/
- https://docs.opensignlabs.com/docs/self-host/docker/run-locally

Installer Docker

- créer un conteneur LXC et installer Docker : https://siocours.lycees.nouvelle-aquitaine.pro/doku.php/docker/installationlinux
- renseigner les dépôts

```
# Add Docker's official GPG key:
apt update
apt install -y ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF

apt update
```

- Installer les packages Docker

```
apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Installer Opensign

- s'authentifier au préalable avec son compte Docker

```
docker login
```

```
* exécuter localement OpenSign :
  * télécharger les fichiers **docker-compose.yml**, **Caddyfile** et **.env.local\_dev**
  * renommer .env.local\_dev en .env.prod
```

```
curl --remote-name-all https://raw.githubusercontent.com/OpenSignLabs/OpenSign/main/docker-compose.yml
https://raw.githubusercontent.com/OpenSignLabs/OpenSign/main/Caddyfile
https://raw.githubusercontent.com/OpenSignLabs/OpenSign/main/.env.local_dev && mv .env.local_dev
.env.prod && docker compose up --force-recreate
```

- Accéder à l'URL https://localhost:3001 et suivre les instructions pour finaliser l'installation.

utiliser un certificat autosigné

- générer un certificat auto-signé incluant l'adresse IP dans le SAN
- configurer Caddy pour l'utiliser
- afin que https://IP:3001 fonctionne sans ERR*SSL*PROTOCOL_ERROR

Générer un certificat auto-signé AVEC SAN contenant l'IP

Il faut IMPÉRATIVEMENT ajouter l'adresse IP dans le champ Subject Alternative Name (SAN), sinon tous les navigateurs refuseront la connexion.

Remplacer X.X.X.X par l'IP du serveur.

```
openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -  -keyout server.key \
  -   -out server.crt \
  -   -subj "/CN=X.X.X.X" \
  -   -addext "subjectAltName = IP:X.X.X.X"
```

- deux fichiers sont obtenus:
    - server.crt → certificat
    - server.key → clé privée
- vérifier le contenu du certificat

```
openssl x509 -in server.crt -text -noout
```

- Placer les fichiers dans un dossier, par exemple **/opt/opensign/certs/**

Importer le certificat dans Caddy

Dans le fichier docker-compose.yml, il faut monter le dossier des certificats et indiquer à Caddy de ne PAS utiliser Let's Encrypt (car tu n'utilises pas un domaine).

Exemple :

```
services:
  caddy:
      image: caddy:latest
      ports:
          - "3001:3001"
      volumes:
          - ./Caddyfile:/etc/caddy/Caddyfile
          - /opt/opensign/certs:/certs
      depends_on:
          - client
          - server
```

Configurer Caddy pour utiliser le certificat auto-signé

- éditer le fichier Caddyfile :

```
https://X.X.X.X:3001 {
    tls /certs/server.crt /certs/server.key
    reverse_proxy /api/* server:8080
    reverse_proxy client:3000
}
```

- Remplace X.X.X.X par l'IP du serveur
- server et client = noms des conteneurs Docker OpenSign (à adapter selon ton docker-compose)

Redémarrer Caddy + OpenSign

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

Tester l'accès externe

Dans le navigateur : https://IP_DU_SERVEUR:3001

Installer mimio - S3

Objectif

Avoir un service S3-compatible prêt à l'emploi (local/on-prem), accessible via https://s3.example.fr avec console d'admin, comptes IAM (users), buckets, politiques, et sauvegardes.

Prérequis

- conteneur Debian 13
  - CPU : 2 vCPU (4+ si charge), RAM : 4–8 Go, Disque : 100+ Go selon besoins
  - Réseau : IP fixe + DNS correct
  - Stockage : idéalement un disque dédié (virtio-scsi, cache write-back) pour des performances stables
- Ports à ouvrir (Firewall Proxmox / VM / routeur) :
  - Mono-instance : 9000/tcp (API S3), 9001/tcp (console) — ou proxifiés derrière 80/443
  - Distribué : ajouter 9000-900X pour chaque nœud et ports internes (TCP)
- Nom de domaine (optionnel mais recommandé) :
  - s3.example.fr pour l'endpoint S3
  - console.s3.example.fr (ou /console) pour la console

Astuce : Dans Proxmox, attacher un disque virtuel distinct pour /data MinIO (facile à migrer/monitorer).

```
docker run -p 9000:9000 -p 9001:9001 \
  -v /minio/data:/data \
  -e "MINIO_ROOT_USER=admin" \
  -e "MINIO_ROOT_PASSWORD=motdepasse" \
  quay.io/minio/minio server /data --console-address ":9001"
```

- Console MinIO : http://IP:9001
- Endpoint S3 : http://IP:9000