

1 - Présentation et architecture

Comprendre Immich, ses composants, ses besoins et les données à protéger.

- [1.1 - Rôle, fonctions et limites d'Immich](#)
- [1.2 - Architecture et services](#)
- [1.3 - Prérequis et dimensionnement](#)
- [1.4 - Données critiques et arborescence](#)

1.1 - Rôle, fonctions et limites d'Immich

Immich est une plateforme auto-hébergée destinée à sauvegarder, parcourir, rechercher et partager des photos et vidéos. Elle fournit une interface Web, des applications mobiles, une chronologie, des albums, la reconnaissance faciale, la recherche intelligente, une carte et des fonctions de partage.

Ce qu'Immich stocke

- les fichiers originaux envoyés depuis le Web, le mobile ou la CLI ;
- les miniatures et aperçus générés ;
- les vidéos transcodées ;
- les images de profil ;
- les sauvegardes automatiques de PostgreSQL ;
- les métadonnées dans PostgreSQL : utilisateurs, chemins, albums, visages, favoris, partages et paramètres.

Deux types de bibliothèques

Type	Fonctionnement
Bibliothèque interne	Immich reçoit et organise les fichiers envoyés par les utilisateurs
Bibliothèque externe	Immich indexe des fichiers déjà présents dans un dossier monté

Une bibliothèque externe ne transforme pas automatiquement un simple dossier en sauvegarde Immich. Les métadonnées ajoutées dans Immich peuvent rester uniquement dans PostgreSQL. Un déplacement de fichier externe peut être interprété comme une suppression puis un nouvel ajout.

Limites à connaître

- Immich ne remplace pas une stratégie de sauvegarde 3-2-1.
- Une sauvegarde PostgreSQL seule ne contient aucune photo ni vidéo.
- Copier uniquement les médias ne préserve pas les albums, utilisateurs, visages et partages.
- Immich ne rescane pas sa bibliothèque interne pour reconstruire automatiquement toute la base.
- Un retour vers une version plus ancienne du serveur n'est pas pris en charge.
- Le serveur et les applications mobiles doivent rester sur des versions majeures compatibles.
- Les fichiers de la bibliothèque interne ne doivent pas être modifiés directement sur le disque.

Bon usage

1. Envoyer les nouvelles photos avec l'application mobile, le Web ou la CLI.
2. Utiliser une bibliothèque externe pour consulter une photothèque existante.
3. Sauvegarder ensemble PostgreSQL, les fichiers Immich et la configuration.
4. Tester régulièrement une restauration sur une instance isolée.
5. Lire les notes de version avant chaque mise à jour majeure.

La dernière version stable vérifiée lors de la rédaction est Immich v3.1.0. Une installation plus ancienne, comme v2.7.5, doit être sauvegardée et migrée en respectant les changements de version.

Références :

- <https://docs.immich.app/>
- <https://docs.immich.app/administration/backup-and-restore/>
- <https://github.com/immich-app/immich/releases/latest>

1.2 - Architecture et services

Une installation Immich standard repose sur plusieurs services. Le nom exact des conteneurs varie entre Docker Compose, TrueNAS et les versions du catalogue.

Composants principaux

Composant	Rôle
<code>immich-server</code>	interface Web, API, uploads et traitements en arrière-plan
PostgreSQL	utilisateurs, métadonnées, albums, chemins et index vectoriels
VectorChord	extension PostgreSQL utilisée pour la recherche vectorielle
Redis	file d'attente et coordination des tâches
<code>immich-machine-learning</code>	recherche intelligente et reconnaissance faciale
Stockage <code>/data</code>	originaux, miniatures, profils, vidéos encodées et sauvegardes

Les anciennes installations peuvent utiliser `pgvector.rs` et des services nommés `pgvector` ou `pgvector_upgrade`. Les versions récentes utilisent l'image PostgreSQL Immich avec VectorChord.

Flux simplifié

1. Le mobile ou le navigateur envoie un fichier au serveur.
2. Le serveur écrit l'original dans le stockage.
3. Le serveur enregistre le chemin et les métadonnées dans PostgreSQL.
4. Redis distribue les tâches en arrière-plan.
5. Les workers extraient les métadonnées, créent les miniatures et transcodent les vidéos.
6. Le service de machine learning calcule les informations de recherche et de visages.

Workers

Le conteneur serveur comprend généralement :

- `api` pour les requêtes Web et mobiles ;
- `microservices` pour les miniatures, vidéos et autres tâches.

Sur une petite installation, les deux restent dans le même service. Les séparer est réservé aux besoins avancés de répartition ou de limitation des ressources.

Vérifier les services Docker

```
docker compose ps
docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Status}}'
docker compose logs --tail=100
```

Sur TrueNAS avec accès au shell :

```
sudo docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Status}}' | grep -i immich
```

Ne pas supposer un nom de conteneur. L'identifier avant d'utiliser `docker logs`, `docker exec` ou `pg_dump`.

Dépendances critiques

Immich ne peut pas fonctionner correctement si :

- PostgreSQL est indisponible ou incompatible ;
- les dossiers de `/data` sont absents ou non accessibles ;
- Redis ne peut pas traiter les files d'attente ;
- le reverse proxy bloque les gros envois ;
- une migration de base a été interrompue.

Références :

- <https://docs.immich.app/developer/architecture/>
- <https://docs.immich.app/administration/jobs-workers/>

1.3 - Prérequis et dimensionnement

Configuration minimale et recommandée

La documentation Immich indique actuellement :

Ressource	Minimum	Recommandation de départ
CPU	2 coeurs	4 coeurs ou davantage
Mémoire	6 Go	8 Go ou davantage
Système	Linux 64 bits	Debian, Ubuntu, TrueNAS ou autre Linux pris en charge
Base PostgreSQL	stockage local	SSD recommandé

Le machine learning, les imports massifs et la génération de miniatures peuvent nécessiter davantage de mémoire et de CPU. Sur TrueNAS, prévoir plus de 8 Go de mémoire pour l'application si le machine learning est activé.

Stockage

Prévoir de l'espace pour :

- les fichiers originaux ;
- les miniatures et aperçus ;
- les vidéos encodées ;
- PostgreSQL ;
- les modèles de machine learning ;
- les sauvegardes locales ;
- la marge nécessaire pendant une migration.

La taille générée dépend fortement des vidéos et des paramètres. Conserver une marge libre suffisante et surveiller séparément l'espace et les inodes.

```
df -hT
df -i
du -sh /chemin/immich/*
```

Réseau

- réseau local stable entre le mobile, le serveur et le stockage ;
- accès HTTPS pour l'extérieur ;
- délais suffisants dans le reverse proxy ;
- upload maximal adapté aux longues vidéos ;
- DNS local et public cohérent si un domaine est utilisé.

GPU

Le GPU est optionnel. Il peut accélérer :

- le transcodage vidéo avec NVENC, Quick Sync ou VAAPI ;
- le machine learning avec CUDA, OpenVINO, ROCm ou un autre moteur pris en charge.

La Quadro P2000 prend en charge NVENC et possède une capacité de calcul CUDA suffisante pour Immich. Le serveur doit néanmoins disposer du pilote NVIDIA officiel et, pour Docker Compose, du NVIDIA Container Toolkit.

Virtualisation

Immich fonctionne dans une machine virtuelle complète. Docker dans un conteneur LXC n'est pas recommandé par le projet pour un déploiement standard. Pour une migration Proxmox, préférer une VM Linux ou déplacer directement Immich vers l'application TrueNAS.

Liste de préparation

- ☐ processeur et mémoire suffisants ;
- ☐ stockage de la base sur SSD si possible ;

- ☐ capacité pour les originaux et les sauvegardes ;
- ☐ heure et DNS corrects ;
- ☐ sauvegarde distante disponible ;
- ☐ accès administrateur au système et à Immich ;
- ☐ versions source et destination identifiées.

Référence : <https://docs.immich.app/install/requirements/>

1.4 - Données critiques et arborescence

Immich utilise six dossiers principaux dans son emplacement de médias, généralement monté dans le conteneur sous `/data`.

Arborescence

Dossier	Contenu	Criticité
<code>upload</code>	originaux envoyés lorsque le modèle de stockage est désactivé	critique
<code>library</code>	originaux organisés lorsque le modèle de stockage est activé	critique si utilisé
<code>profile</code>	images de profil	critique
<code>thumbs</code>	miniatures et aperçus générés	régénérable
<code>encoded-video</code>	vidéos transcodées	régénérable
<code>backups</code>	sauvegardes automatiques de PostgreSQL	critique avec les médias

Exemple Docker Compose :

```
UPLOAD_LOCATION=/srv/immich/data
DB_DATA_LOCATION=/srv/immich/postgres
```

Exemple TrueNAS adapté à ton environnement :

```
/mnt/HDD_DATA_TRUENAS/IMMICH/data
/mnt/HDD_DATA_TRUENAS/IMMICH/pgData
```

Ces sous-dossiers sont une organisation recommandée. Vérifier l'installation existante avant de créer ou déplacer quoi que ce soit.

La base ne contient pas les photos

PostgreSQL contient les chemins, métadonnées et relations. Une sauvegarde `.sql.gz` ne contient pas les originaux. Une restauration complète exige donc :

1. une sauvegarde PostgreSQL compatible ;
2. une copie cohérente des dossiers de médias ;
3. la configuration et les montages ;
4. les bibliothèques externes accessibles aux mêmes chemins vus par le conteneur.

Bibliothèque externe

Une bibliothèque externe doit être montée sous un chemin distinct, par exemple :

```
Hôte TrueNAS : /mnt/HDD_DATA_TRUENAS/PHOTOS
Conteneur    : /mnt/external-libraries/photos
```

Le chemin saisi dans l'interface Immich est celui du conteneur :

```
/mnt/external-libraries/photos
```

Identifier les chemins réels

Docker Compose :

```
grep -E '^(UPLOAD_LOCATION|DB_DATA_LOCATION|IMMICH_VERSION)= ' .env
docker inspect immich_server --format '{{json .Mounts}}'
```

TrueNAS :

```
sudo docker inspect NOM_CONTENEUR_IMMICH --format '{{json .Mounts}}'
sudo findmnt | grep -i immich
```

Ne pas modifier directement les fichiers générés sous `/mnt/.ix-apps/app_configs/`. TrueNAS peut les recréer lors d'une mise à jour de l'application.

Références :

- <https://docs.immich.app/administration/backup-and-restore/>

- <https://docs.immich.app/install/truenas/>