260614 - procedure Joplin-compression

Contexte — instructions de Julien (260614)

J'utilise l'application Joplin sur tous mes appareils. C'est ma mémoire de tout (travail, hobbies, maison...) et donc à force, la base de données se remplit. Elle se compose de notes à l'intérieur desquelles on trouve des images. Même en n'important que des captures d'écran, le poids de ces images est devenu important — de l'ordre de 300 Mo initialement estimé (476 Mo constatés en réalité, voir ci-dessous).

Appareils synchronisés sous Joplin :

- VPS Juxjux.OVH (serveur de sync)
- PC Nexte (PC du travail de Julien)
- PC Elio+Jux (PC maison de Julien)
- Mobile Xiaomi T15pro de Julien
- Tablette Samsung S5 de Julien
- Session Ubuntu sur clé SSD de Julien

Objectif de la procédure :

1. Compresser le stock d'images contenues dans la base de données Joplin via un process fiable de remplacement
2. Monter un système de compression automatique quotidienne des nouvelles images

Principes définis par Julien :

1. Tenir un carnet de bord (observatoire) du poids digital de la BDD Joplin dans la page BookStack Joplin, avec les 15 pièces jointes les plus lourdes
2. Mettre en place sur le VPS une procédure duplication → compression → remplacement d'images, qui se propage via la sync Joplin sur tous les appareils

---

Exploration technique — Claude, 14/06/2026

Infrastructure Joplin sur le VPS

Contrairement à ce qu'on pourrait attendre, Joplin Server **n'utilise pas SQLite** mais **PostgreSQL**.

Container	Image	Rôle
`joplin`	`joplin/server:latest`	Serveur de sync Joplin
`joplin-db`	`postgres:15-alpine`	Base de données
`joplin-nginx`	`nginx:alpine`	Reverse proxy interne
`joplin_to_obsidian`	image custom	Container migration (actif, sans impact)

Volume de données : `/home/debian/joplin-data` → `/home/joplin/.config/joplin` (bind mount)

Connexion DB : `POSTGRES_HOST=joplin-db`, `POSTGRES_DATABASE=joplin`, `POSTGRES_USER=joplin`

Important : le port 5432 de joplin-db n'est **pas exposé** à l'extérieur du réseau Docker. Tout script de manipulation doit tourner sur le VPS et se connecter via l'IP interne Docker.

Structure de la base de données

La table centrale est `items` (23 tables au total). Chaque note, ressource et paramètre Joplin est une ligne dans cette table.

Colonnes clés :

- `content` (bytea) — données binaires brutes
- `content_size` (integer) — taille en octets
- `content_storage_id` = 1 → stockage de type `Database` (tout est dans PostgreSQL, pas de fichiers externes)
- `jop_type` — type d'item Joplin
- `updated_time` — timestamp de dernière modification (utilisé par les clients pour détecter les changements à sync)

Répartition par type :

jop_type	Signification	Nombre	Poids total
0	Ressource (image/fichier joint)	736	**476 MB**
1	Note	699	8,7 MB
4	Tag	733	4,5 MB
13	NoteTag (relation note↔tag)	250	3,9 MB
2	Carnet (Folder)	94	707 KB
6	Master Key	87	19 KB
5	Setting	39	—

Analyse des ressources (jop_type = 0)

Les ressources sont stockées comme **bytes bruts** dans la colonne `content`. Le format est détectable via les magic bytes :

- **PNG** (`\x89PNG`) — majorité des ressources
- **JPEG** (`\xff\xd8`) — portion significative
- **ZIP** (`PK\x03\x04`) — cas particulier : ZIP contenant plusieurs images `page_1.png`, `page_2.png`... (documents multi-pages)

La colonne `mime_type` est vide pour toutes les ressources — le type est implicite dans les bytes du contenu.

Top 15 ressources les plus lourdes (état initial) :

Rang	ID	Taille	Format
1	0xh87pWrRt82z0DbO9n3yQ	12,2 MB	ZIP (page_1.png 7MB + page_2.png 5MB)
2	M0C32hKC2hMqPAxpRGiVgp	7,2 MB	PNG
3	UbAqQY3cEbH62vioIHU6Pj	6,0 MB	PNG
4	ZPQJVSjptMSxE71pV7JbAt	5,9 MB	PNG
5	h4Lhw1Trx9wgmD7doX9NyZ	5,9 MB	PNG
6	sbktcNb4IUj0yIovgDt0fL	5,3 MB	PNG
7	jtzuGpvrRTR12iLzwhcSNn	5,2 MB	PNG
8	VasIoF2e9EGuNQt38aOFMx	4,9 MB	JPEG

Rang	ID	Taille	Format
9	ZACDcQWzQzDLdnV7Qnf933	4,3 MB	PNG
10	hExoJEEWT2tHz1demE5Nhm	4,3 MB	PNG
11	veu4HT3bStx07gRUlAEYvG	4,2 MB	PNG
12	bz9Twmb2F5lj0mPIQi48IB	4,2 MB	JPEG
13	TzWK21r4n0yvbEtJh02DGB	4,2 MB	JPEG
14	eD165w1bdEHJg0tq3qWok5	4,2 MB	PNG
15	HD4SeqIx252nP9nh8HDVnH	4,1 MB	PNG

Architecture retenue

Choix de compression — décision Julien, 14/06/2026

PNG → JPEG 85% (lossy). Toutes les images, qu'elles soient PNG ou JPEG à l'origine, sont converties/re-sauvegardées en JPEG qualité 85. Gain estimé : 50–75% par image. Acceptable pour des captures d'écran.

Pour les ZIP multi-pages : chaque image interne est convertie en JPEG 85%, le ZIP est reconstruit.

Phase 1 — Observatoire

Script `joplin_observatoire.py` sur le VPS :

- Connexion psycopg2 à joplin-db via IP réseau Docker interne
- Calcul des stats globales (taille totale, nombre d'items par format)
- Liste des 15 ressources les plus lourdes avec format et taille
- Mise à jour de la page BookStack Joplin (page 166) avec ces informations

Phase 2 — Compression (script principal)

Script `joplin_compress.py` sur le VPS :

Connexion : psycopg2 → IP Docker interne de `joplin-db` : 5432

Traitement par ressource :

1. Lire le blob `content` depuis `items` (jop_type=0)
2. Détecter le format (magic bytes)
3. Ouvrir avec Pillow, convertir en JPEG 85 (`quality=85, optimize=True`)
4. Pour les ZIP multi-pages : dézipper → compresser chaque image → reconstruire le ZIP
5. Si le gain est > 5% : mettre à jour `content`, `content_size`, `updated_time` en base
6. Logger le résultat (ID, taille avant, taille après, ratio)

Tracking des items traités : fichier JSON local `/home/debian/joplin_compress_log.json` — évite de retraiter les ressources déjà compressées lors des passages quotidiens.

Propagation sync : Joplin détecte les changements via `updated_time`. Lors de la prochaine synchronisation de chaque client, les ressources compressées sont re-téléchargées automatiquement.

Phase 3 — Service systemd (cron quotidien)

Timer systemd `joplin-compress.timer` → `joplin-compress.service` :

- Déclenchement quotidien (3h du matin)
- Traite uniquement les nouvelles ressources (non présentes dans le log JSON)
- Met à jour l'observatoire BookStack après chaque passage

Mise en production — 14/06/2026

Résultat du premier run (stock complet)

Ressources traitées	736 au total
Compressées	424
Ignorées (gain < 5%)	312
Erreurs	0
Poids avant	476 Mo

Poids après	119,7 Mo
Économie	**356 Mo (-78,9%)**

Scripts déployés sur le VPS

Script	Rôle	Options
`/home/debian/joplin_compress.py`	Compression Pillow PNG/JPEG → JPEG 85%, ZIP multi-pages, mise à jour PostgreSQL	`--dry-run` (simulation) / `--limit N` (N ressources max)
`/home/debian/joplin_observatoire.py`	Stats DB + top 15 → mise à jour page BookStack Joplin (ID 166)	—

Log tracking : `/home/debian/joplin_compress_log.json` — liste des ressources déjà traitées, évite les doublons aux runs suivants.

Timer systemd

Service	`joplin-compress.service`
Timer	`joplin-compress.timer`
Déclenchement	Chaque nuit à 3h UTC (`OnCalendar=*-*-* 03:00:00`)
Logs	`/var/log/joplin-compress.log`
Statut	active (waiting) — prochain run : 15/06/2026 03:00 UTC

Notes techniques

- joplin-db IP Docker interne : `172.27.0.4` (peut changer si le container est recréé — vérifier avec `docker inspect joplin-db -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'`)
- La compression modifie `content`, `content_size` et `updated_time` dans la table `items`
- Les clients Joplin re-téléchargent les ressources modifiées lors de la prochaine synchronisation (détection via `updated_time`)
- Les images à fond transparent (RGBA/LA/P) sont aplaties sur fond blanc avant conversion JPEG

Revision #9
Created 2026-06-14 08:32:10 UTC by Julien
Updated 2026-06-14 12:58:51 UTC by Julien