

260827-Tofs à compresse sur le PC Windows

On dépose des photos dans un dossier, on les récupère compressées dans un autre. Rien à lancer, rien à régler au coup par coup. C'est le principe de la procédure [Komga-PDF](#) appliqué aux images, mais **entièrement sur le PC** `julie` — le VPS n'intervient pas.

Dépôt	<code>D:\Procedures photos\Tofs a compresse\</code>
Résultat	<code>D:\Procedures photos\Tofs-compressed\</code>
Moteur	<code>caesiumclt</code> v1.4.0 — binaire Rust, aucune dépendance
Script	<code>Jux-scripts/Photos-Caesium/compress_photos.py</code> — stdlib seule
Profil actif	<code>q80</code> — environ -62 % , sans redimensionnement
Délai	43 secondes entre le dépôt et la livraison

La seule chose à ne jamais toucher : l'option `-e` dans `options communes` de `config.json`. Elle conserve l'EXIF, et ce n'est **pas** le comportement par défaut de Caesium. Sans elle, dates de prise de vue et coordonnées GPS sont purement supprimées — silencieusement. Ce n'est pas une précaution théorique : c'est déjà arrivé, le 14 juillet 2026, et 275 photos y ont laissé leur GPS définitivement.

Au quotidien

Déposez les photos dans `Tofs a compresse\` et **n'attendez rien d'autre** : la tâche `Photos-Caesium` tourne en permanence et les prend d'elle-même. Les fichiers compressés apparaissent dans `Tofs-compressed\` avec la même arborescence — un album déposé revient en album.

```
20:43:51  dépôt dans Tofs a compresse\  
20:44:32  déclenchement (après 20 s sans changement)  
20:44:34  livraison dans Tofs-compressed\
```

Le délai tient au garde-fou : la surveillance attend **20 secondes sans le moindre changement** avant de traiter, ce qui évite d'attraper un fichier en cours de copie. Si vous videz une carte SD de 300 photos, elle patiente jusqu'à la fin du transfert.

Ce qui devient de vos originaux :

- ils partent dans `originaux-traites\` — **jamais effacés**, c'est le dossier à vider de temps en temps ;
- un HEIC ou un RAW, que Caesium ne sait pas lire, est écarté dans `non-traites\` plutôt que traité ;
- une photo qui ne gagnerait rien à être compressée est **recopiée telle quelle** en sortie : un fichier déposé ressort toujours.

À la main

Trois lanceurs à double-cliquer dans `D:\Procedures photos\` :

Lanceur	Effet
<code>Compresser maintenant.cmd</code>	une passe, puis sortie
<code>Simuler le gain.cmd</code>	compresse en zone tampon, annonce le gain, jette le résultat
<code>Surveillance (fenetre visible).cmd</code>	la boucle, console à l'écran

```
py -3.12 compress_photos.py --une-fois
py -3.12 compress_photos.py --simulation
py -3.12 compress_photos.py --profil sans-perte --une-fois
py -3.12 compress_photos.py --profils
py -3.12 compress_photos.py                # surveillance continue
```

La surveillance automatique

Tâche planifiée `Photos-Caesium` (ONLOGON), en place depuis le 2026-08-27 :

```
schtasks /Create /TN "Photos-Caesium" /TR
'"C:\Users\julie\AppData\Local\Programs\Python\Python312\pythonw.exe"
"D:\Syncthing\Jux_univers\Jux-scripts\Photos-Caesium\compress_photos.py"' /SC ONLOGON /RL
LIMITED /F
```

`pythonw.exe` et non `python.exe`, sans quoi une console s'ouvre à chaque ouverture de session. Vérification par `schtasks /Query /TN "Photos-Caesium" /V /FO LIST` ou `taskschd.msc` ; retrait par `/Delete`.

Cette tâche doit être créée depuis la console de Julien, jamais depuis une session Claude Code. Une session Claude tourne dans un conteneur MSIX : la tâche figerait des chemins virtualisés et ne ferait rien, sans le moindre message. Voir la [page 282](#).

Profils de compression

Définis dans `config.json`, modifiables sans toucher au code. `--profil <nom>` surcharge ponctuellement.

Profil	Options	Gain sur photos de téléphone
<code>q80</code> (actif)	<code>-q 80</code>	-59 à -72 %
<code>q90</code>	<code>-q 90</code>	environ -45 %
<code>sans-perte</code>	<code>--lossless</code>	-23 %
<code>web-3000px</code>	<code>-q 80 --long-edge 3000 --no-upscale</code>	environ -85 %

Options communes à tous les profils : `-e --keep-dates --min-savings 1%`.

Le profil actif ne redimensionne pas. `q80` ne touche ni aux dimensions ni au nombre d'étiquettes EXIF. `web-3000px`, qui ramènerait le grand côté à 3 000 pixels, figure dans `config.json` mais **n'est pas sélectionné** — à n'activer que sur demande explicite.

« Sans altérer la qualité » coûte les deux tiers du gain

C'était la demande initiale, et elle mérite d'être précisée : les deux modes de Caesium n'ont rien de comparable en rendement.

- `--lossless` **ne touche pas un seul pixel.** Il reconstruit les tables de Huffman et réoptimise le conteneur. Sur 71 photos réelles : **-23 %**.
- `-q 80` **ré-encode l'image.** La différence est invisible à l'œil sur une photographie, et c'est le réglage par défaut de l'interface graphique Caesium — celui qui donne l'impression que « Caesium compresse admirablement bien ». Sur les mêmes fichiers : **-64 %**.

D'où `q80` par défaut. Pour des originaux irremplaçables, basculer sur `sans-perte` dans `config.json`, ou traiter ce lot-là avec `--profil sans-perte`.

Ce qui est garanti : EXIF, GPS, dates, dimensions

Vérifié deux fois, par la mesure et non par relecture du code.

Contrôle du 2026-09-28 — la chaîne réelle, de bout en bout

Trois photos du Xiaomi 15T Pro, dont deux géolocalisées, **déposées sans rien lancer** : c'est la surveillance qui les a prises, comme pour un dépôt ordinaire.

Contrôle	Résultat
Bloc EXIF	identique à l'octet près — 52 334 / 52 740 / 52 782 octets
Coordonnées GPS	43.675676, 4.634059 avant et après, au chiffre près
Date de prise de vue	identique à la seconde
Étiquettes EXIF	72 → 72 (58 → 58 pour la photo sans GPS)
Dimensions	4096×3072 inchangées
Date de fichier (<code>mtime</code>)	identique — <code>--keep-dates</code> tient
Poids	13,1 Mo → 5,1 Mo, -61 %

Les 72 étiquettes conservées couvrent tout ce que l'appareil écrit : marque, modèle, objectif, ouverture, temps de pose, ISO, orientation, balance des blancs, et le bloc GPS complet. **Rien n'est aplati ni simplifié** — Caesium ne réécrit que les données d'image.

Mesure de référence du 2026-08-27

71 photos de téléphone (lot « Ligurie 2026 »), Ryzen 5 2600X, `--threads 0` :

	Taille	Durée
Originaux	251,1 Mo	—
<code>q80</code>	90,4 Mo (-64,0 %)	22 s
<code>sans-perte</code>	-22,8 %	—

Soit environ **12 Mo/s** de débit : un vidage de carte SD de 8 Go se traite en une dizaine de minutes. Le contrôle EXIF de ce jour-là portait sur trois fichiers et donnait déjà un bloc EXIF identique à l'octet (1 819 / 52 665 / 52 597 octets), IFD GPS présent, dimensions inchangées.

L'outil de contrôle

`Jux-scripts/Photos-Caesium/comparer_exif.py` — `stdlib` seule. Prend deux dossiers et compare fichier par fichier : bloc EXIF octet par octet, GPS converti en degrés décimaux, `DateTimeOriginal`, appareil, dimensions, nombre d'étiquettes, `mtime`, poids.

```
py -3.12 comparer_exif.py <dossier_reference> <dossier_sortie>
```

Il rend un **verdict global**, et se prête au contrôle de santé : lancé avec le même dossier des deux côtés, il doit tout déclarer conforme. À rejouer après toute modification de profil, de `config.json` ou du binaire.

⚠ L'incident du 2026-07-14 — le - e manquant

Une passe antérieure à la mise en place de cette procédure a été lancée **sans -e**.

Dégâts constatés le 2026-08-28, sur les photos de l'été 2026 : **275 fichiers dépouillés de tout bloc EXIF** — date et GPS. Signature nette : 493 Ko de moyenne pour les fichiers traités, contre 1 103 Ko pour les 76 intacts du même voyage.

Conséquence différée, invisible sur le moment. Sans `DateTimeOriginal`, Immich se rabat sur la date du fichier. Or le WebDAV kDrive ne préserve pas les `mtime` : à la première resynchronisation, six semaines plus tard, les 275 photos sont allées se ranger au jour de la synchro et la chronologie du voyage s'est effondrée.

Le GPS est définitivement perdu. Seule la date a pu être reconstruite, depuis le nom des fichiers, par `dater_photos_nas.py`. Récit complet : [page 164](#).

À faire avant toute passe de compression : vérifier `-e` dans la ligne de commande, et contrôler l'EXIF d'un fichier de sortie avant de traiter le lot. Un fichier de sortie sans bloc EXIF doit faire arrêter la passe immédiatement — les originaux ne sont pas toujours récupérables.

PC et NAS : le même moteur, une garantie de plus sur le NAS

Le stock photo du NAS suit sa propre procédure, en quatre étapes avec validation humaine ([page 285](#)). Le moteur, lui, est rigoureusement le même.

	PC (<code>compress_photos.py</code>)	NAS (<code>traiter_lot_nas.py</code>)
Binaire	<code>caesiumclt</code> 1.4.0, identique	
Profil	<code>-q 80</code> , identique	
Options communes	<code>-e --keep-dates --min-savings 1%</code> , identiques	
Date et propriétaire réimposés puis revérifiés	non	oui (<code>os.utime</code> , <code>os.chown</code> , puis relecture)

La différence ne porte pas sur l'EXIF, protégé à l'identique par `-e` des deux côtés. Elle porte sur la date de fichier et le propriétaire, pour une raison propre au NAS : les photos y appartiennent à `admin:users`, et une compression lancée sans privilèges les réattribuerait toutes en cassant SMB et Synology Photos. Sur le PC il n'y a qu'un utilisateur, et la mesure montre que `--keep-dates` suffit.

Formats

Pris en charge : JPEG, PNG, WebP, GIF, TIFF.

Non pris en charge : HEIC (iPhone) et RAW. Ces fichiers partent dans `non-traites\` — **jamais supprimés** — avec une ligne au journal. Si le besoin HEIC se présente, il faudra une étape de conversion en amont : Caesium ne sait pas le lire.

Arborescence

D:\Procedures photos\		
├ Tofs a compresse\	← dépôt (sous-dossiers acceptés)	
├ Tofs-compressed\	← résultat, même arborescence	

─ originaux-traites\	← originaux après traitement
─ non-traites\	← formats non pris en charge
─ echecs\	← abandons après 3 tentatives
─ bin\caesiumclt.exe	
─ logs\photos_compress.log	
─ config.json	
─ Compresser maintenant.cmd	
─ Simuler le gain.cmd	
─ Surveillance (fenetre visible).cmd	

Le script, lui, vit dans `Jux-scripts/Photos-Caesium/`, donc dans l'arbre Syncthing : versionné et disponible sur toutes les machines. **Seules les données restent locales.**

Pourquoi les dossiers sont hors de Syncthing

Ils avaient d'abord été créés dans `D:\Syncthing\`, puis déplacés. **Ne pas les y remettre.**

Il n'existe qu'un seul dossier Syncthing dans tout le maillage (`afltj-njyuy`), partagé avec 7 appareils dont le Xiaomi et la tablette SM-T720. Tout ce qui y tombe se réplique partout. C'est supportable pour Komga-PDF — un PDF de 130 Mo de temps en temps — mais un vidage de carte SD descendrait intégralement sur les téléphones, *avant même* que la compression ait commencé.

Conséquence assumée : **le dépôt à distance n'est pas possible**, on dépose depuis le PC.

Maintenance

- **Vider** `originaux-traites\` **de temps en temps** — les originaux y sont conservés, pas effacés. C'est le dossier qui grossit.
- **Surveiller** `non-traites\` — s'il se remplit, des HEIC ou des RAW arrivent dans le flux.
- `echecs\` **doit rester vide**. Toute entrée signale un fichier corrompu ou un cas non prévu : consulter le journal.
- **Journal** : `logs\photos_compress.log`, rotation automatique à 5 Mo.

Reste à faire : décider du sort du lot de démonstration laissé dans `Tofs-compressed\Ligurie 2026\` (71 photos, 90 Mo), qui sert à juger la qualité `q80` à l'œil.

Annexe — fonctionnement interne

1. **Sondage toutes les 15 s** du dossier de dépôt. Il n'y a pas d'notify sous Windows ; le sondage est de toute façon la partie robuste du dispositif.
2. **Déclenchement après 20 s sans le moindre changement** dans l'arborescence — taille ou date d'un fichier quelconque. Plafond de sécurité à 15 min, au-delà duquel on traite quand même.
3. Le lot entier part en **un seul appel** à `caesiumclt`, qui parallélise sur tous les cœurs (`--threads 0`).
4. **Écriture d'abord dans** `.staging`, puis déplacement : jamais de fichier partiel visible dans `Tofs-compressed\`.
5. Structure préservée (`-R -S`).
6. Collision de nom en sortie : suffixe `(2)`, `(3)`... **Rien n'est jamais écrasé.**
7. Gain inférieur à 1 % (`--min-savings 1%`) : Caesium n'écrit rien, le script recopie l'original tel quel.
8. **Mémoire des échecs** : 3 tentatives (clé taille + date, dans `.etat.json`), puis mise à l'écart dans `echecs\`.
9. Journal persistant avec rotation à 5 Mo.

Les points 2, 4, 7, 8 et 9 reprennent d'emblée les correctifs apportés à `komga_watch_vps.sh` le 2026-08-23 ([page 220](#)) : événements perdus pendant le traitement, écriture non atomique, absence de mémoire des échecs, journal volatil. Autant ne pas refaire les mêmes erreurs.

`config.json`, clé `apres_traitement` : `deplacer` (défaut), `supprimer`, ou `laisser`. **laisser fait retraiter le dépôt en boucle** à chaque passe — à réserver aux tests.

Annexe — pièges rencontrés à l'installation

1. `--dry-run` **de Caesium ne mesure rien.** L'option existe et s'exécute sans erreur, mais le rapport JSON renvoie `compressed_size == original_size`, soit 0,0 % de gain : elle simule l'écriture, pas la compression. Le mode `--simulation` du script compresse donc réellement dans `.staging`, relève le chiffre, puis jette le résultat.
2. `-e` **n'est pas le défaut** — perte silencieuse des dates et du GPS. Voir l'incident ci-dessus.
3. `--json` **est la bonne interface de pilotage.** Caesium renvoie un rapport structuré, une entrée par fichier (`original_path`, `compressed_size`, `status`...) plus un `summary`. Le script s'appuie dessus plutôt que de parser la sortie texte.

4. `--min-savings` **rend inutile le garde-fou maison.** Sur Komga-PDF il avait fallu écrire à la main la règle « si la compression ne gagne rien, recopier l'original » ; Caesium l'implémente nativement.
5. **Créer la tâche planifiée depuis Claude Code : à éviter** (piège MSIX). La création avait d'ailleurs été refusée par le garde-fou de session — ce qui tombait bien.
6. **Le binaire ne s'installe pas.** `caesiumclt.exe` est un exécutable Rust statique de 5,7 Mo, sans dépendance ni installateur : on le dépose dans `bin\` et on le met à jour en le remplaçant.

Annexe — le binaire

Version	<code>caesiumclt</code> v1.4.0, publiée le 8 juillet 2026
Source	github.com/Lymphatus/caesium-clt/releases
Archive	<code>caesiumclt-v1.4.0-x86_64-pc-windows-msvc.zip</code> — 2 296 760 octets
SHA-256	<code>a56454a83207fc25830f4d12679d7385c0906060f2ce5f54345ac5a4cf94b00f</code>
Licence	GPL / MPL selon les composants (<code>bin\LICENSE.md</code>)

Le même dépôt publie `caesiumclt-v1.4.0-x86_64-unknown-linux-musl.tar.gz`, statique — c'est lui qui tourne sur le NAS, sans Docker ni Entware.

Annexe — pourquoi 100 % local, contrairement à Komga-PDF

Komga-PDF passe par le VPS. Photos-Caesium non, et c'est délibéré. Trois raisons, par ordre d'importance :

1. **Il n'y a pas de service à réutiliser.** Komga-PDF s'appuie sur StirlingPDF, déjà déployé et qui expose `compress-pdf` en HTTP. Pour les images, aucun équivalent : Caesium est un binaire en ligne de commande, pas un service. L'héberger sur le VPS n'apporterait pas une API, juste un exécutable — qu'on peut aussi bien poser sur le PC.
2. **Le CPU.** La compression JPEG est du calcul pur, massivement parallélisable. Le Ryzen 5 2600X offre 12 threads et ne fait généralement rien ; le VPS fait tourner une trentaine de conteneurs, avec un swap déjà occupé et un antécédent de saturation disque à 100 %.

3. **Le transfert.** Les photos sont déjà sur place. Les envoyer au VPS pour les faire revenir consommerait deux fois le débit montant de la Livebox sans contrepartie.

Le seul cas qui justifierait le VPS serait de pouvoir déposer depuis le téléphone pendant que le PC est éteint — c'est précisément ce qui avait fait choisir le VPS pour Komga-PDF. Le portage serait trivial (binaire linux-musl statique, script en stdlib seule). À reconsidérer si le besoin apparaît.

Voir aussi

- [Page 285](#) — les photos du NAS : procédure en quatre étapes avec validation humaine
- [Page 220](#) — Komga-PDF, dont celle-ci reprend le principe et les correctifs
- [Page 164](#) — Immich : réparation des dates après l'incident du `-e` manquant
- [Page 282](#) — le piège MSIX de Claude Code sur le poste `julie`

Copie de référence de cette page : `Jux-scripts/Photos-Caesium/doc_bookstack_284.md`. Mise en place le 2026-08-27, page réécrite intégralement le 2026-09-28.

Revision #7

Created 2026-08-27 17:36:30 UTC by Julien

Updated 2026-09-28 20:10:52 UTC by Julien