

17_Komga

komga

Image	gotson/komga
Version	1.25.0
Etat	Up 9 hours
Compose project	komga
Reseau	komga_default
URL	https://komga.juxjux.ovh
Source	https://github.com/gotson/komga

Ports

Hote	Container	IP
25600	25600/tcp	0.0.0.0
25600	25600/tcp	::

Volumes

Source (hote)	Destination	Type	Mode
/home/debian/docker/komga/config	/config	bind	rw
/home/debian/komga	/data	bind	rw
/etc/timezone	/etc/timezone	bind	ro
	/tmp	volume	rw

Integration Claude Code — MCP

Statut MCP	MCP actif
Outils (prefix)	mcp__komga__
Configuration	Script Python : C:\Users\eliob\.claude\mcp_komga.py Auth : Basic Auth (Base64 login:password)

Outils disponibles

- get_libraries
- get_series
- get_series_detail
- get_series_books
- get_book
- get_book_metadata
- get_books
- get_latest_books
- get_new_series
- get_updated_series
- get_on_deck
- get_read_lists
- get_read_list
- get_collections
- get_collection
- search
- mark_series_read
- update_book_progress
- get_recently_added

Ce que nous pouvons faire ensemble

- Parcourir la bibliothèque comics/manga par série et tome
 - Voir les dernières acquisitions et séries mises à jour
 - Marquer une série comme lue ou mettre à jour la progression
 - Gérer les listes de lecture et collections
 - Rechercher dans toute la bibliothèque
-

Lenteur d'affichage des pages PDF — diagnostic et correctif (2026-08-30)

Symptome : temps d'attente de 4 a 6 secondes a chaque page d'un PDF, y compris sur une page déjà consultée. Décourageant a l'usage.

Ce qui n'est PAS en cause : la definition de rendu

Premier reflexe : abaisser la definition des pages. **Ce n'est pas possible, et cela n'aurait pratiquement rien change.**

La resolution de rasterisation des PDF est **cablee en dur** dans Komga, dans

`StaticConfiguration.kt` :

```
@Bean("pdfResolution") fun pdfResolution(): Float = 3200F
@Bean("pdfImageType") fun pdfImageType() = ImageType.JPEG
```

`PdfExtractor` calcule `scale = 3200 / min(largeur, hauteur)`, soit **3200 px sur le petit cote** : une page A4 sort en $3200 \times 4040 = \mathbf{12,9\text{ Mpx}}$, 0,75 a 1,4 Mo de JPEG.

Aucune propriete `komga.pdf.*` n'existe dans les metadonnees de configuration Spring du jar (verifie sur la 1.25.0 installée), et la documentation officielle n'expose rien de tel.

Attention — deux variables inertes dans la stack Portainer 7 : `KOMGA_PDF_RENDERING_DPI=72` et `KOMGA_IMAGE_QUALITY=LOW` ne correspondent a aucune propriete existante. Spring les ignore en silence. Elles laissent croire a un réglage actif : **ne pas s'y fier**. Laissees en place le 2026-08-30 sur decision de Julien.

La vraie cause : le montage rclone etait en cache d'écriture seule

`kdrive-komga.service` tournait en `--vfs-cache-mode writes`, qui ne met en cache **que les écritures**. Chaque lecture repartait vers kDrive, indefiniment. Or PDFBox saute d'abord a la table xref en fin

de fichier, puis picore les objets : plusieurs allers-retours reseau par page.

C'etait le seul montage kDrive reste en `writes` : music, livres, audiobooks, photo et romans etaient deja en `full` avec un plafond de 10G. Une exception oubliee, pas un choix.

Mesures — ou passait reellement le temps

Livre temoin : *Geo 2012 - Iles de Reve.pdf*, 96,8 Mo, 140 pages.

Operation	Resultat
Page rendue en JPEG	12,9 Mpx — 4,5 a 6,0 s
Page PDF brute (aucun rendu)	60 a 90 Ko — 3,8 a 4,2 s
Acces aleatoire de 1 Mo sur le montage	0,52 a 0,67 s, identique a la 2e lecture
Lecture sequentielle du fichier entier	84 a 111 Mo/s (97 Mo en ~1 s)

Le plancher de ~3,9 s etait **commun au rendu et a la page brute** : la preuve que le coupable etait l'I/O, pas la rasterisation, qui n'ajoutait que 0,6 a 2 s.

Consequence contre-intuitive : lire un fichier de 100 Mo de bout en bout (~1 s) coute **moins cher** que d'y picorer trois pages en acces aleatoire (~2 s). C'est le principe du prechauffage decrit plus bas.

Correctif applique le 2026-08-30

Options ajoutees a l'`ExecStart` de `kdrive-komga.service` (sauvegarde : `kdrive-komga.service.bak-260830`) :

```
--vfs-cache-mode full --vfs-cache-max-size 8G --vfs-cache-max-age 6h
--vfs-cache-poll-interval 1m --vfs-read-chunk-size 32M
--vfs-read-chunk-size-limit 512M
```

	Avant (<code>writes</code>)	Apres (<code>full</code>)
1re page d'un livre froid	4,5 a 6,0 s	3,4 s
Pages suivantes	4,5 a 6,0 s	0,69 a 0,84 s
Meme page redemandee	6,0 s (aucun gain)	0,99 s

Facteur 6 a 7 sur tout ce qui suit la premiere page. Les 0,7 s residuels sont la rasterisation PDFBox : le plancher est desormais le CPU du VPS, plus rien a voir avec kDrive.

Pieges a ne pas refaire

- **Redemarrer le container Komga apres le remontage** — il garde sinon l'ancienne reference FUSE et continue de lire dans le vide. Meme piege que le correctif `:shared` du 2026-06-08.
- **Ne jamais passer un mount en `full` sans `--vfs-cache-max-size`**. Six montages en `full` totalisent 58 Go de plafond theorique pour 34 Go libres sur `/home/debian`. En pratique le cache reel plafonne vers 4,3 Go, car `--vfs-cache-max-age` purge bien avant que la taille ne morde — mais l'incident du 2026-06-24 (spool rclone de 14 Go, disque plein, VPS a terre) rappelle ce qui arrive sans bornage.
- **Le choix est l'inverse de celui retenu pour Jellyfin (`minimal`)**, et a juste titre : un film est lu une fois en flux lineaire, un PDF subit des acces aleatoires repetes sur le meme fichier.
- L'API renvoie le chemin *vu par le container* (`/data/...`) : le retraduire en `/home/debian/komga/...` cote hote.

Piste non deployee : le prechauffage a l'ouverture

`--vfs-cache-mode full` telecharge a la demande, pas d'un bloc. Un demon peut forcer la lecture sequentielle complete des l'ouverture du livre, ce qui supprime les 3,4 s de la premiere page.

Ecrit et teste, **non installe** (le cache seul suffisait) : `Jux-scripts/Komga-Cache/precharger_komga.py` + `komga-precharge.service`, avec un `README.md` detailant mesures, deploiement et retour arriere.

Declencheur retenu : `GET /api/v1/books/{id}/manifest` dans le journal nginx — appel emis par le lecteur web **avant la premiere page** (verifie dans le JS du webreader). Repli sur `pages/1-3` pour les clients tiers. La suppression du fichier est deleguee a rclone : `--vfs-cache-max-age` vaut mieux qu'une suppression a la fermeture, car un livre rouvert dix minutes plus tard reste chaud.

Retour arriere

```
sudo cp /etc/systemd/system/kdrive-komga.service.bak-260830 \
      /etc/systemd/system/kdrive-komga.service
sudo systemctl daemon-reload
sudo systemctl restart kdrive-komga.service
sudo docker restart komga
```

Note : la section « Integration Claude Code — MCP » ci-dessus est perimee. Le script `C:\Users\eliob\.claude\mcp_komga.py` a disparu avec le profil Windows `eliob` ; il a ete reecrit sans

dependance dans `Jux-scripts/MCP/mcp_komga.py`, mais n'est pas enregistre sur le poste `julie` (seuls `bookstack` et `kdrive` le sont).

Nouveaux fichiers invisibles de la bibliotheque — 2026-09-02

Symptome : un magazine depose (Beaux Arts de septembre 2026) n'apparait pas dans Komga, malgre plusieurs scans manuels.

Cause : le cache de listing du montage, `--dir-cache-time 72h`

`rclone` conserve la **liste des fichiers d'un dossier** pendant 72 h sans jamais la reverifier. Un fichier ajoute sur kDrive apres le dernier listing reste donc invisible du montage — et donc de Komga — pendant **jusqu'a trois jours**. Aucun scan ne peut y remedier : Komga ne voit que ce que le systeme de fichiers lui montre.

Preuve relevee le 2026-09-02 sur `Arts/Beaux Arts/Beaux Arts 2026` :

Vue	Contenu
Le montage (<code>ls</code>)	5 fichiers , listing fige au 5 mai
kDrive en direct (<code>rclone lsl</code>)	7 fichiers , dont deux deposes le 01/09 a 14:21

Ce n'etait pas un seul magazine : la purge du cache a fait apparaitre **20 livres** d'un coup — Revue du Vin de France (juin, juillet-aout, septembre), Saveurs, Monde Gourmand × 3, Elle a Table, Cuisine et Vins, Connaissance des Arts × 4, Destination Portugal/Italie/Espagne, Beaux Arts × 3. Des mois de nouveautes.

Fausses pistes — ne pas les reprendre

- **Ce n'est ni Komga ni son scan.** Les taches de scan s'executaient normalement toutes les nuits (cron `0 4 * * * ?`), sans la moindre erreur au journal. Relancer un scan ne pouvait rien changer.
- **Ce n'est pas la synchro NAS → kDrive.** Elle fonctionnait parfaitement : `There was nothing to transfer`, 36 608 fichiers verifiees le 02/09 a 00:01. Le fichier etait bien monte du NAS vers kDrive le 01/09 a 16:21. Ne pas rejouer le diagnostic de la panne du 2026-08-

28 (`rcclone.conf` deplace) : le canari `grep -l CRITICAL logs/sync_*` etait muet.

- **Piege de diagnostic rencontre en direct** : un `rcclone lsl --include '**eaux*rts**' | tail -12` n'a retourné que des fichiers de 2025, ce qui a fait conclure à tort que le fichier était absent de kDrive. Le filtre remontait en fait le sous-dossier *Hors Series*, et `tail` coupait le reste. **Toujours lister le dossier exact, sans filtre glob ni troncature**, avant de conclure à l'absence d'un fichier.

Reparation immediate

`kdrive-komga.service` **n'a pas de flag `--rc`** : il est le seul montage rclone du VPS qu'on **ne peut pas** rafraichir par `rcclone rc vfs/refresh`. Le seul moyen de purger son cache de listing est donc de redemarrer le service.

```
sudo systemctl restart kdrive-komga.service
ls "/home/debian/komga/Arts/Beaux Arts/Beaux Arts 2026/" # le fichier apparait
sudo docker restart komga                               # ancienne ref FUSE
# puis scan de chaque bibliotheque :
curl -u USER:PASS -X POST http://127.0.0.1:25600/api/v1/libraries/{id}/scan
```

Le cache VFS de *donnees* (les 8 Go sur disque) **survit** au redemarrage du service : seul le cache de listing, qui vit en memoire, est perdu. Rien n'est retelecharge.

Deux pieges rencontres pendant la reparation

- **ERR_1005 transitoire**. Sur les 20 livres, 2 sont sortis du scan en `ERROR`. Les fichiers étaient sains (`%PDF-1.5 ... %%EOF`) : c'est l'analyse de 20 PDF *simultanement sur un montage encore froid* qui a fait echouer deux lectures. Une simple relance (`POST /api/v1/books/{id}/analyze`) les a passés en `READY`. **Ne pas conclure a un PDF corrompu sans avoir reanalyse une fois.**
- **La recherche Komga par titre est trompeuse**. `?search=Beaux Arts - 09` ne renvoyait pas le livre alors qu'il était bien indexé et `READY` : la tokenisation Lucene ne matche pas ce genre de titre. **Vérifier la présence d'un livre par `?library_id={id}`, jamais par `?search=`.**

Correctif durable — APPLIQUE le 2026-09-02

Ne pas baisser `--dir-cache-time` : sur un dossier de cette taille, une valeur courte provoque une instabilité permanente (kDrive ne répond pas assez vite entre deux expirations, le dossier apparaît vide). 72 h reste le bon réglage — ce qui manquait, c'était la *capacité à rafraîchir à la demande*.

Trois actions, appliquées le 2026-09-02 :

1. **Flag** `--rc` **ajoute au montage** — il en était dépourvu, seul de tous les montages du VPS (sauvegarde de l'unit : `kdrive-komga.service.bak-260902`) :

```
--rc --rc-addr 127.0.0.1:5574 --rc-user=rcadmin --rc-pass=RcKomga2026!
```

Le rafraîchissement du listing prend **18 s** pour 1 580 dossiers et 38 187 fichiers, sans redémarrer quoi que ce soit :

```
rclone rc --rc-addr 127.0.0.1:5574 --rc-user=rcadmin --rc-pass='RcKomga2026!' \
  vfs/refresh recursive=true
```

2. **Règle ufw** `DENY 5574/tcp` **ajoutée**. Les ports 5572, 5573, 5575, 5576 et 5577 étaient bloqués depuis l'incident du 2026-06-09, mais **pas 5574** — il n'existait pas encore. Les six sont désormais couverts uniformément. L'écoute reste sur `127.0.0.1` de toute façon.
3. **Cron** `45 3 * * *` → `Jux-scripts/Komga-Cache/komga_scan.sh` : rafraîchit le listing puis relance les 23 bibliothèques, **avant** le scan interne de Komga à 4 h, pour que celui-ci en profite aussi. Journal : `/home/debian/logs/komga_scan.log`. Modèle repris de `navidrome_scan.sh`, en production depuis juin 2026.

Exécution de contrôle : `vfs/refresh` 18 s, 23 scans lancés, 24 s au total, code de sortie 0.

Rafraîchissement immédiat, quand un fichier vient d'être déposé et qu'on ne veut pas attendre la nuit :

```
bash /home/debian/Documents/Jux_univers/Jux-scripts/Komga-Cache/komga_scan.sh
```

Les identifiants du script vivent dans `/home/debian/.komga_scan.env` (chmod 600), **hors de l'arbre Syncthing** ; le script lui-même est versionné dans `Jux-scripts/` et se réplique sur les 7 appareils.

Note annexe : Syncthing sur le poste `julie` n'avait pas détecté seul le nouveau script — les deux instances s'annonçaient `idle` avec 0 fichier en attente. Il a fallu forcer le scan par `POST /rest/db/scan?folder=afltj-njyuy&sub=...`. À garder en tête avant de conclure à une panne de synchronisation.

Revision #6

Created 2026-05-29 09:41:34 UTC by Julien

Updated 2026-09-02 13:27:55 UTC by Julien