h06-documents-organization

- 00-chapter-intro.md
- 01-paperless-ngx.md
- 02-bookstack.md
- 03-karakeep.md
- 04-memos.md
- 05-booklore.md
- 06-mealie.md
- 07-homebox.md
- 08-medikeep.md
- 09-lubelogger.md
- 10-hortusfox.md
- 11-linkstack.md
- 12-reitti.md
- Swarm-Reitti Bridge
- AdventureLog - Travel Helper

00-chapter-intro.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Chapter Introduction tags: [documents, organization, paperless, bookstack, productivity]

Overview

This chapter covers personal productivity, knowledge management, and life-organisation services. These range from document archival and wiki systems through to recipe management, home inventory, plant care, and fitness tracking.

Services in This Chapter

Service	Container(s)	Purpose
Paperless-NGX	`paperless-webserver-1`, `paperless-db-1`, `paperless-broker-1`, `paperless_gotenberg`, `paperless_tika`	Document management and archival
BookStack	`bookstack`, `bookstack-db`	Self-hosted wiki and documentation
Karakeep	`karakeep-web`	Bookmarks and read-it-later
Memos	`memos`	Quick notes and journal

Service	Container(s)	Purpose
Booklore	`booklore`, `booklore-db`	Personal book library and reading tracker
Mealie	`mealie`, `mealieaddons`	Recipe management and meal planning
Homebox	`homebox`	Home inventory management
Medikeep	`medikeep`, `medikeep-db`	Personal medical records tracker
LubeLogger	`lubelogger`, `lubelogger-db`	Vehicle maintenance log
HortusFox	`hortusfox`, `hortusfox-db`, `hortusfox-cron`	Plant care and garden management
LinkStack	`linkstack`	Public link-in-bio / personal landing page
Reitti	`reitti-app`, `reitti-postgis`, `reitti-redis`, `reitti-rabbitmq`, `reitti-tiles`, `reitti-photon`	Activity and fitness tracking
Swarm-Reitti Bridge	`swarm-reitti-bridge`	Custom Foursquare→Reitti check-in sync service

---

Last Updated: 2026-06-17

01-paperless-ngx.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Paperless-NGX tags: [paperless, documents, ocr, archive, productivity]

Overview

Paperless-NGX is the centralised document management and archival system. It ingests scanned documents and PDFs via a watched consume directory and email, applies OCR, tags, correspondents, and document types, then stores the processed results in a structured archive. Gmail OAuth is configured for email ingestion. SSO is provided via Authentik OpenID Connect.

Access

Type	URL	Auth
External	`https://paperless.jeeves5454.ddns.net`	Authentik OIDC + GeoBlock + CrowdSec
Internal	`https://paperless.home.local`	Step-CA TLS (internal)

Authentication uses Authentik OIDC (`openid_connect` Django allauth provider). Regular local login is kept enabled (`PAPERLESS_DISABLE_REGULAR_LOGIN=false`). Auto-signup for new OIDC users is disabled — accounts must be pre-created.

Containers

Five containers in this stack:

Container	Image	Role
`paperless-webserver-1`	`ghcr.io/paperless-ngx/paperless-ngx:latest`	Web UI + workers
`paperless-db-1`	`postgres:16`	Primary database
`paperless-broker-1`	`redis:7`	Celery task queue
`paperless_gotenberg`	`gotenberg/gotenberg:8.27`	DOCX→PDF conversion
`paperless_tika`	`apache/tika:latest`	Content extraction

paperless-webserver-1

Main application container running both the Django web server and Celery workers.

Key environment variables:

Variable	Value / Notes
`PAPERLESS_URL`	`https://paperless.jeeves5454.ddns.net`
`PAPERLESS_CSRF_TRUSTED_ORIGINS`	`https://paperless.jeeves5454.ddns.net`
`PAPERLESS_OAUTH_CALLBACK_BASE_URL`	`https://paperless.jeeves5454.ddns.net`
`PAPERLESS_DBHOST`	`db`
`PAPERLESS_REDIS`	`redis://broker:6379`
`PAPERLESS_TIKA_ENABLED`	`1`
`PAPERLESS_TIKA_ENDPOINT`	`http://tika:9998`
`PAPERLESS_TIKA_GOTENBERG_ENDPOINT`	`http://gotenberg:3000`
`PAPERLESS_TASK_WORKERS`	`2`
`PAPERLESS_THREADS_PER_WORKER`	`2`
`PAPERLESS_TIME_ZONE`	`America/Toronto`
`PAPERLESS_OCR_LANGUAGE`	`eng`
`PAPERLESS_CONSUMER_RECURSIVE`	`true`
`PAPERLESS_CONSUMER_SUBDIRS_AS_TAGS`	`true`
`PAPERLESS_CONSUMER_POLLING`	`5` (seconds)
`PAPERLESS_FILENAME_FORMAT`	`{{ created_year }}/{{ document_type }}/{{ created_year }}-{{ created_month }}-{{ created_day }}_{{ correspondent }}_{{ title }}`

Variable	Value / Notes
`PAPERLESS_FILENAME_FORMAT_REMOVE_NONE`	`true`
`PAPERLESS_SOCIAL_AUTO_SIGNUP`	`false`
`PAPERLESS_ACCOUNT_EMAIL_VERIFICATION`	`none`
`PAPERLESS_GMAIL_OAUTH_CLIENT_ID`	(visible — treat as semi-public)
`PAPERLESS_GMAIL_OAUTH_CLIENT_SECRET`	**REDACTED**
`PAPERLESS_APPS`	`allauth.socialaccount.providers.openid_connect`
OIDC provider client secret	**REDACTED** (inside `PAPERLESS_SOCIALACCOUNT_PROVIDERS`)

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/paperless/data`	`/usr/src/paperless/data`	Index and SQLite state
`/mnt/data/paperless/media`	`/usr/src/paperless/media`	Archived document files
`/mnt/data/paperless/consume`	`/usr/src/paperless/consume`	Watched consume folder
`/mnt/data/paperless/export`	`/usr/src/paperless/export`	Export output folder

> The `media`, `consume`, and `export` directories live on the NFS-equivalent Ceph volume at `/mnt/data/`, providing separation from the system drive.

paperless-db-1

PostgreSQL 16 database backend.

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/paperless/pgdata`	`/var/lib/postgresql/data`

paperless-broker-1

Redis 7 for Celery task queue.

Bind mounts:

Host Path	Container Path

`/home/jeeves/docker/paperless/redis`	`/data`

paperless_gotenberg

Gotenberg 8.27 — converts Office documents (DOCX, XLSX, etc.) to PDF for ingestion. No persistent volumes.

paperless_tika

Apache Tika — extracts content and metadata from complex document formats. No persistent volumes.

Traefik Labels

```
traefik.http.routers.paperless-external.rule: Host(`paperless.jeeves5454.ddns.net`)
traefik.http.routers.paperless-external.entrypoints: websecure
traefik.http.routers.paperless-external.tls.certresolver: letsencrypt
traefik.http.routers.paperless-external.middlewares: plex-geoblock@file,crowdsec-bouncer@file

traefik.http.routers.paperless-internal.rule: Host(`paperless.home.local`)
traefik.http.routers.paperless-internal.entrypoints: websecure
traefik.http.routers.paperless-internal.tls.certresolver: step-ca
```

Notes / Gotchas

- Secrets in `PAPERLESS_SOCIALACCOUNT_PROVIDERS` (OIDC client secret) and `PAPERLESS_GMAIL_OAUTH_CLIENT_SECRET` are in plaintext in the Portainer-managed compose environment — rotate if the Portainer env is ever exposed.
- The consume directory polling interval is 5 seconds. Drop files into `/mnt/data/paperless/consume` (or a subdirectory — subdirectory names become tags automatically via `PAPERLESS_CONSUMER_SUBDIRS_AS_TAGS`).
- Tika and Gotenberg must be running for complex document formats to process; plain-text PDFs work without them.
- The `PAPERLESS_LOGOUT_REDIRECT_URL` is set to the Authentik end-session endpoint — clicking Log Out in Paperless also terminates the Authentik session.

Last Updated: 2026-06-17

02-bookstack.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: BookStack tags: [bookstack, wiki, documentation, mariadb, oidc]

Overview

BookStack is the self-hosted personal knowledge base and documentation wiki — the platform this very documentation is written in. It organises content into Shelves → Books → Chapters → Pages and provides a full Markdown editor. Access is protected by Authentik OIDC on the external route. On the internal route, authentication still passes through BookStack's own OIDC flow (no Traefik ForwardAuth bypass), so the Authentik session is always required.

Access

Type	URL	Auth
External	`https://wiki.jeeves5454.ddns.net`	Authentik OIDC + GeoBlock + CrowdSec
Internal	`https://wiki.home.local`	Authentik OIDC (Step-CA TLS)

BookStack uses its own OIDC integration rather than Traefik ForwardAuth. The external Traefik route applies GeoBlock and CrowdSec but **not** `authentik-auth@docker` — BookStack manages the OIDC redirect itself.

Containers

Container	Image	Role
`bookstack`	`lscr.io/linuxserver/bookstack:latest`	Web application
`bookstack-db`	`mariadb:10.11`	MariaDB database

bookstack (application)

Key environment variables:

Variable	Value / Notes
`APP_URL`	`https://wiki.jeeves5454.ddns.net`
`AUTH_METHOD`	`oidc`
`AUTH_AUTO_INITIATE`	`true` — skips BookStack login page, redirects to Authentik
`OIDC_NAME`	`Authentik`
`OIDC_ISSUER`	`https://auth.jeevesconsults.ca/application/o/book-stack-website-s/`
`OIDC_ISSUER_DISCOVER`	`true`
`OIDC_CLIENT_ID`	`QjpMMmpDqNCQIM75np6gGyMkN9Y569AtyNVhfJvx`
`OIDC_CLIENT_SECRET`	**REDACTED**
`OIDC_EXTERNAL_ID_CLAIM`	`email`
`OIDC_DISPLAY_NAME_CLAIMS`	`name`
`OIDC_FETCH_AVATAR`	`false`
`OIDC_END_SESSION_ENDPOINT`	`false`
`DB_HOST`	`bookstack-db`
`DB_DATABASE`	`bookstack`
`DB_USERNAME`	`bookstack`
`DB_PASSWORD`	**REDACTED**
`MAIL_HOST`	`smtp.gmail.com`
`MAIL_PORT`	`587`
`MAIL_USERNAME`	`jeeves5454@gmail.com`
`MAIL_ENCRYPTION`	`TLS`
`MAIL_FROM`	`noreply@jeevesconsults.ca`
`PUID` / `PGID`	`1000`
`TZ`	`America/Toronto`

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/bookstack/config`	`/config`	App config, attachments, uploads
`/home/jeeves/docker/bookstack/public`	`/public`	Public web assets

bookstack-db (MariaDB 10.11)

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/bookstack/db`	`/var/lib/mysql`

Traefik Labels

```
traefik.http.routers.bookstack-ext.rule: Host(`wiki.jeeves5454.ddns.net`)
traefik.http.routers.bookstack-ext.entrypoints: websecure
traefik.http.routers.bookstack-ext.tls.certresolver: letsencrypt
traefik.http.routers.bookstack-ext.middlewares: plex-geoblock@file,crowdsec-bouncer@file
traefik.http.routers.bookstack-ext.service: bookstack-svc

traefik.http.routers.bookstack-int.rule: Host(`wiki.home.local`)
traefik.http.routers.bookstack-int.entrypoints: websecure
traefik.http.routers.bookstack-int.tls.certresolver: step-ca
traefik.http.routers.bookstack-int.service: bookstack-svc

traefik.http.services.bookstack-svc.loadbalancer.server.port: 80
```

Notes / Gotchas

- `APP_URL` is set to the external URL even though both internal and external routes exist. This is intentional — it is used for OIDC redirect URIs and email links, which must be reachable externally.
- `AUTH_AUTO_INITIATE: true` means anyone who hits the URL is immediately redirected to Authentik. There is no BookStack login form shown unless OIDC fails. To bypass OIDC in an emergency, use `?prevent_auto_init=true` appended to the URL.

- For an existing user to link their Authentik account, go to Admin → Users → select user → set **External Authentication ID** to their email address.
- The LSIO image uses `DB_USERNAME`, not `DB_USER`. Using `DB_USER` silently falls back to no authentication and causes confusing login failures.
- `APP_KEY` (Laravel application key) is stored in the `/config` directory and persists across container restarts. Do not delete the config bind mount or the key must be regenerated and all sessions will be invalidated.

Last Updated: 2026-06-17

03-karakeep.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Karakeep tags: [karakeep, bookmarks, read-later, productivity]

Overview

Karakeep is a self-hosted bookmark manager and read-it-later service. It crawls saved URLs for their full text and screenshots, indexes them via Meilisearch, and supports OCR on images (`ocr_langs=eng`). Video download is enabled (yt-dlp backend, no size limit). SMTP is configured for email-to-save via Gmail. New signups are disabled — account management is manual.

Access

Type	URL	Auth
External	`https://bookmark.jeevesconsults.ca`	GeoBlock + CrowdSec (own auth)

External-only — no internal `home.local` route configured.

Configuration

Image: `ghcr.io/karakeep-app/karakeep:latest`

Key Environment Variables

Variable	Value / Notes
`NEXTAUTH_URL`	`https://bookmark.jeevesconsults.ca`
`NEXTAUTH_URL_INTERNAL`	`http://karakeep-web:3000`
`NEXTAUTH_TRUST_HOST`	`true`
`NEXTAUTH_SECRET`	**REDACTED**
`DATA_DIR`	`/data`
`ASSETS_DIR`	`/assets`
`MEILI_ADDR`	`http://meilisearch:7700`
`MEILI_MASTER_KEY`	**REDACTED**
`BROWSER_WEB_URL`	`http://chrome:9222` (headless Chromium)
`OCR_LANGS`	`eng`
`OCR_CONFIDENCE_THRESHOLD`	`75`
`CRAWLER_VIDEO_DOWNLOAD`	`true`
`CRAWLER_VIDEO_DOWNLOAD_MAX_SIZE`	`-1` (unlimited)
`CRAWLER_VIDEO_DOWNLOAD_TIMEOUT_SEC`	`7200`
`INFERENCE_ENABLE_AUTO_SUMMARIZATION`	`true`
`MAX_ASSET_SIZE_MB`	`50`
`DISABLE_SIGNUPS`	`true`
`EMAIL_VERIFICATION_REQUIRED`	`false`
`SMTP_HOST`	`smtp.gmail.com`
`SMTP_PORT`	`587`
`SMTP_SECURE`	`true`
`SMTP_USER`	`jeeves5454@gmail.com`
`SMTP_FROM`	`jeeves5454@gmail.com`
`SMTP_PASSWORD`	**REDACTED**

Traefik Labels

```
traefik.http.routers.karakeep.rule: Host(`bookmark.jeevesconsults.ca`)
traefik.http.routers.karakeep.entrypoints: websecure
traefik.http.routers.karakeep.tls.certresolver: letsencrypt
traefik.http.routers.karakeep.middlewares: plex-geoblock@file,crowdsec-bouncer@file,karakeep-
```

```
headers
traefik.http.middlewares.karakeep-headers.headers.customrequestheaders.X-Forwarded-Proto:
https
traefik.http.services.karakeep.loadbalancer.server.port: 3000
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/karakeep/data`	`/data`	Database and app data
`/mnt/data/karakeep`	`/assets`	Saved assets (screenshots, videos)

Stack Companions

Karakeep requires additional sidecar containers (Meilisearch, Headless Chrome) in the same compose network to function. These are not independently accessible.

Notes / Gotchas

- The `X-Forwarded-Proto: https` middleware is required for Karakeep (Next.js) to trust the proxy and generate correct redirect URIs — without it, HTTPS callbacks will fail.
- Video downloads can consume significant storage on `/mnt/data/karakeep`. The timeout is 2 hours, which accommodates long-form content.
- Meilisearch index data is stored in `/data` — this must be backed up alongside the app database for search to remain functional after a restore.
- `DISABLE_SIGNUPS: true` — new accounts must be created manually in the admin UI.

Last Updated: 2026-06-17

04-memos.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Memos tags: [memos, notes, journal, productivity]

Overview

Memos is a lightweight, self-hosted note-taking and micro-journaling app. It provides a simple Twitter/Mastodon-style feed of notes, supports Markdown, tags, and public/private visibility per memo. Internal-only access — no external route is configured.

Access

Type	URL	Auth
Internal	`https://memos.home.local`	Memos own auth (Step-CA TLS)

No external route — accessible only on the LAN.

Configuration

Image: `neosmemo/memos:stable`

Environment Variables

Variable	Value	Purpose
`MEMOS_INSTANCE_URL`	`https://memos.home.local`	Canonical URL
`MEMOS_PORT`	`5230`	Listen port
`TZ`	`America/Toronto`	Timezone

Traefik Labels

```
traefik.http.routers.memos.rule: Host(`memos.home.local`)
traefik.http.routers.memos.entrypoints: websecure
traefik.http.routers.memos.tls.certresolver: step-ca
traefik.http.services.memos.loadbalancer.server.port: 5230
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/memos/data`	`/var/opt/memos`	Database and assets

Memos uses SQLite stored in `/var/opt/memos`. No external database dependency.

Notes / Gotchas

- `neosmemo/memos:stable` tracks the stable release tag. Upgrading the image will migrate the SQLite database automatically on first startup.
- There is no Authentik integration on this service — Memos manages its own accounts internally.
- Memos supports public content sharing via short links; ensure private memos are set to private if personal content is being stored.

Last Updated: 2026-06-17

05-booklore.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Booklore tags: [booklore, books, library, mariadb, step-ca]

Overview

Booklore is a personal e-book library manager. It indexes books from the NFS Multimedia share (read-only) and a local bookdrop directory for new additions. The app is a Java/Spring Boot application (JDK 25 with Shenandoah GC) backed by MariaDB. It uses its own internal authentication — no Authentik ForwardAuth or OIDC. The Step-CA root certificate is bind-mounted into the system trust store to allow Booklore to make HTTPS calls to internal `*.home.local` services.

Access

Type	URL	Auth
External	`https://booklore.jeevesconsults.ca`	GeoBlock + CrowdSec (own auth)
Internal	`https://booklore.home.local`	Own auth (Step-CA TLS)

Containers

Container	Image	Role
`booklore`	`grimmory/grimmory:latest`	Web application
`booklore-db`	`lscr.io/linuxserver/mariadb:latest`	MariaDB database

booklore (application)

Key environment variables:

Variable	Value / Notes
`DATABASE_USERNAME`	`booklore`
`DATABASE_PASSWORD`	**REDACTED**
`DATABASE_URL`	`jdbc:mariadb://booklore-db:3306/booklore`
`BOOKLORE_PORT`	`6060`
`DISK_TYPE`	`NETWORK` (books from NFS mount)
`USER_ID` / `GROUP_ID`	`1000`
`TZ`	`America/Toronto`
`APP_VERSION`	`v3.2.0`

Runtime: Java 25 (Temurin JDK 25.0.3) with Shenandoah GC.
JVM tuning: max 60% RAM, Shenandoah compact heuristics, 256MB metaspace cap.

Bind mounts:

Host Path	Container Path	Purpose
`/mnt/Multimedia/Books`	`/books`	Book library (read-only via NFS)
`/home/jeeves/docker/booklore/data`	`/app/data`	App state and metadata
`/home/jeeves/docker/booklore/bookdrop`	`/bookdrop`	Drop zone for new books
`/home/jeeves/docker/step-ca/config/certs/root_ca.crt`	`/usr/local/share/ca-certificates/step-ca.crt`	Step-CA root trust injection

booklore-db (MariaDB LSIO)

Image: `lscr.io/linuxserver/mariadb:latest`

Variable	Value
`MYSQL_DATABASE`	`booklore`
`MYSQL_USER`	`booklore`

Variable	Value
`MYSQL_PASSWORD`	**REDACTED**
`PUID` / `PGID`	`1000`

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/booklore/mariadb`	`/config`

Traefik Labels

```
traefik.http.routers.booklore-external.rule: Host(`booklore.jeevesconsults.ca`)
traefik.http.routers.booklore-external.entrypoints: websecure
traefik.http.routers.booklore-external.tls.certresolver: letsencrypt
traefik.http.routers.booklore-external.middlewares: plex-geoblock@file,crowdsec-bouncer@file
traefik.http.routers.booklore-external.service: booklore-svc

traefik.http.routers.booklore-internal.rule: Host(`booklore.home.local`)
traefik.http.routers.booklore-internal.entrypoints: websecure
traefik.http.routers.booklore-internal.tls.certresolver: step-ca
traefik.http.routers.booklore-internal.service: booklore-svc

traefik.http.services.booklore-svc.loadbalancer.server.port: 6060
```

Notes / Gotchas

- The Step-CA root certificate bind mount (`root_ca.crt` → `/usr/local/share/ca-certificates/`) makes the container's Java runtime trust internal TLS certificates. This is required if Booklore makes any HTTPS requests to `*.home.local` services.
- `/mnt/Multimedia/Books` is the NFS share from UnRAID — it must be mounted before Booklore starts or the library scan will fail silently. Check `df -h /mnt/Multimedia` if the library appears empty.
- Drop new e-books (EPUB, PDF) into `/home/jeeves/docker/booklore/bookdrop` for automatic library import.
- `grimmory/grimmory:latest` is the container image name for the Booklore project.

Last Updated: 2026-06-17

06-mealie.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Mealie tags: [mealie, recipes, meal-planning, oidc, authentik]

Overview

Mealie is a self-hosted recipe manager and meal planner. It imports recipes from URLs, supports manual entry, and provides a weekly meal plan interface. OIDC authentication is via Authentik, with auto-redirect enabled. The `mealieaddons` sidecar (Mealie Addons by razziel89) provides enhanced recipe retrieval, PDF export via Pandoc, and image embedding.

Access

Type	URL	Auth
External	`https://recipe.jeevesconsults.ca`	Authentik OIDC + GeoBlock + CrowdSec
Internal	`https://mealie.home.local`	Authentik OIDC (Step-CA TLS)

OIDC auto-redirect is enabled (`OIDC_AUTO_REDIRECT: true`) — the Mealie login form is bypassed and Authentik is shown directly.

Containers

Container	Image	Role
`mealie`	`ghcr.io/mealie-recipes/mealie:latest`	Core application
`mealieaddons`	`ghcr.io/razziel89/mealie-addons:latest`	Enhanced retrieval and export

mealie (application)

Key environment variables:

Variable	Value / Notes
`BASE_URL`	`https://recipe.jeevesconsults.ca`
`OIDC_AUTH_ENABLED`	`true`
`OIDC_AUTO_REDIRECT`	`true`
`OIDC_PROVIDER_NAME`	`Authentik`
`OIDC_CONFIGURATION_URL`	`https://auth.jeevesconsults.ca/application/o/mealie/.well-known/openid-configuration`
`OIDC_CLIENT_ID`	`cjTjv6CLnu0pT7qkkthDTjXmaVPYdxMfOKW9d5Wy`
`OIDC_CLIENT_SECRET`	**REDACTED**
`OIDC_USER_GROUP`	`family-friends`
`OIDC_ADMIN_GROUP`	`admins`
`OIDC_SIGNUP_ENABLED`	`false`
`OIDC_REMEMBER_ME`	`true`
`ALLOW_SIGNUP`	`false`
`SMTP_HOST`	`smtp.gmail.com`
`SMTP_PORT`	`587`
`SMTP_AUTH_STRATEGY`	`TLS`
`SMTP_USER`	`jeeves5454@gmail.com`
`SMTP_PASSWORD`	**REDACTED**
`SMTP_FROM_EMAIL`	`jeeves5454@gmail.com`
`PUID` / `PGID`	`1000`
`TZ`	`America/Toronto`

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/mealie/data`	`/app/data`	Database and assets

mealieaddons

Mealie Addons provides enhanced recipe scraping, Pandoc-based PDF/EPUB export, and image embedding for recipes.

Key environment variables:

Variable	Value / Notes
`MEALIE_BASE_URL`	`https://recipe.jeevesconsults.ca`
`MEALIE_RETRIEVAL_URL`	`http://mealie:9000`
`MA_SELF_URL`	`http://localhost:9000`
`MA_LISTEN_INTERFACE`	`:9000`
`MA_IMAGE_ACTION`	`embed`
`MA_TIMEOUT_SECS`	`60`
`MA_RETRIEVAL_LIMIT`	`5`
`GIN_MODE`	`release`
`PANDOC_FLAGS`	`--epub-title-page=false`

No bind mounts — stateless.

Traefik route:

```
traefik.http.routers.mealieaddons.rule: Host(`mealieaddons.home.local`)
traefik.http.routers.mealieaddons.entrypoints: websecure
traefik.http.routers.mealieaddons.tls.certresolver: step-ca
traefik.http.services.mealieaddons.loadbalancer.server.port: 9000
```

Traefik Labels (mealie)

```
traefik.http.routers.mealie-ext.rule: Host(`recipe.jeevesconsults.ca`)
traefik.http.routers.mealie-ext.entrypoints: websecure
traefik.http.routers.mealie-ext.tls.certresolver: letsencrypt
traefik.http.routers.mealie-ext.middlewares: plex-geoblock@file,crowdsec-bouncer@file,mealie-
```

```
headers
traefik.http.routers.mealie-ext.service: mealie-svc

traefik.http.routers.mealie-int.rule: Host(`mealie.home.local`)
traefik.http.routers.mealie-int.entrypoints: websecure
traefik.http.routers.mealie-int.tls.certresolver: step-ca
traefik.http.routers.mealie-int.service: mealie-svc

traefik.http.middlewares.mealie-headers.headers.customrequestheaders.X-Forwarded-Proto: https
traefik.http.services.mealie-svc.loadbalancer.server.port: 9000
```

Notes / Gotchas

- `OIDC_USER_GROUP: family-friends` limits OIDC login to members of that Authentik group. Users not in this group can be denied access even with valid Authentik credentials. Manage group membership in Authentik admin.
- The `X-Forwarded-Proto: https` header middleware is required for Mealie to generate correct OIDC redirect URIs. Without it, HTTPS URLs will not form correctly and the OIDC flow will fail.
- Mealie uses SQLite by default when no external database is configured. The database file lives in `/app/data`. Back up this directory before upgrades.
- `mealieaddons` is accessible at `mealieaddons.home.local` — this is the URL to configure in Mealie settings for the enhanced importer.

Last Updated: 2026-06-17

07-homebox.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Homebox tags: [homebox, inventory, home-management, authentik]

Overview

Homebox is a home inventory management app for tracking household items, their locations, warranties, and purchase records. The external route uses Authentik ForwardAuth for SSO. The internal route applies `X-Forwarded-Proto` only (Homebox manages its own session internally). Signups are disabled; registration is not allowed.

Access

Type	URL	Auth
External	`https://homebox.jeeves5454.ddns.net`	Authentik ForwardAuth + GeoBlock + CrowdSec
Internal	`https://homebox.home.local`	Homebox own session (Step-CA TLS)

Configuration

Image: `ghcr.io/sysadminsmedia/homebox:latest`

Key Environment Variables

Variable	Value / Notes
`HBOX_MODE`	`production`
`HBOX_LOG_LEVEL`	`info`
`HBOX_LOG_FORMAT`	`text`
`HBOX_WEB_MAX_UPLOAD_SIZE`	`10` (MB)
`HBOX_OPTIONS_ALLOW_REGISTRATION`	`false`
`HBOX_OPTIONS_ALLOW_ANALYTICS`	`false`
`HBOX_STORAGE_PREFIX_PATH`	`data`
`TZ`	`America/Toronto`

Traefik Labels

```
traefik.http.routers.homebox-ext.rule: Host(`homebox.jeeves5454.ddns.net`)
traefik.http.routers.homebox-ext.entrypoints: websecure
traefik.http.routers.homebox-ext.tls.certresolver: letsencrypt
traefik.http.routers.homebox-ext.middlewares: authentik-auth@docker,plex-
geoblock@file,crowdsec-bouncer@file,homebox-headers
traefik.http.routers.homebox-ext.service: homebox-svc

traefik.http.routers.homebox-int.rule: Host(`homebox.home.local`)
traefik.http.routers.homebox-int.entrypoints: websecure
traefik.http.routers.homebox-int.tls.certresolver: step-ca
traefik.http.routers.homebox-int.middlewares: homebox-headers
traefik.http.routers.homebox-int.service: homebox-svc

traefik.http.middlewares.homebox-headers.headers.customrequestheaders.X-Forwarded-Proto: https
traefik.http.services.homebox-svc.loadbalancer.server.port: 7745
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose

`/home/jeeves/docker/homebox/data`	`/data`	SQLite database and uploads

Homebox uses SQLite (`homebox.db`) stored in the `/data` bind mount with WAL journal mode and foreign key enforcement enabled.

Notes / Gotchas

- The external route applies `authentik-auth@docker` ForwardAuth — all external access requires a valid Authentik session. The internal route skips ForwardAuth but applies the `X-Forwarded-Proto` header middleware (required for HTTPS-aware links and session cookies).
- `HBOX_OPTIONS_ALLOW_REGISTRATION: false` disables self-service account creation. New accounts must be created by an admin in the Homebox UI.
- Homebox stores photos and attachments in the `/data` bind mount alongside the SQLite database. Back up the entire directory before upgrades.

Last Updated: 2026-06-17

08-medikeep.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Medikeep tags: [medikeep, medical, health, postgres, authentik-sso]

Overview

Medikeep is a personal medical records tracker. It stores health records, prescriptions, appointments, and related documents. SSO is configured via Authentik (native OIDC integration — not Traefik ForwardAuth). The external route uses GeoBlock and CrowdSec; the internal route is TLS-only with no additional auth middleware.

Access

Type	URL	Auth
External	`https://medical.jeeves5454.ddns.net`	Authentik OIDC + GeoBlock + CrowdSec
Internal	`https://medikeep.home.local`	Authentik OIDC (Step-CA TLS)

SSO is handled by Medikeep's built-in `SSO_ENABLED: true` — it redirects to Authentik on login. Traefik does not apply ForwardAuth to this service.

Containers

Container	Image	Role
`medikeep`	`ghcr.io/afairgiant/medikeep:latest`	Web application
`medikeep-db`	`postgres:15.8-alpine`	PostgreSQL database

medikeep (application)

Key environment variables:

Variable	Value / Notes
`SSO_ENABLED`	`true`
`SSO_PROVIDER_TYPE`	`authentik`
`SSO_ISSUER_URL`	`https://auth.jeevesconsults.ca/application/o/medikeep/`
`SSO_CLIENT_ID`	`EN44sdEMtf29bgTN077W48XsCSpi9bj1Wk0eypI1`
`SSO_CLIENT_SECRET`	**REDACTED**
`SSO_REDIRECT_URI`	`https://medical.jeeves5454.ddns.net/auth/sso/callback`
`DB_HOST`	`medikeep-db`
`DB_PORT`	`5432`
`DB_NAME`	`medical_records`
`DB_USER`	`medapp`
`DB_PASSWORD`	**REDACTED**
`LOG_LEVEL`	`DEBUG`
`LOG_ROTATION_METHOD`	`logrotate`
`ENABLE_API_DOCS`	`false`
`DEBUG`	`false`
`PUID` / `PGID`	`1000`
`TZ`	`America/Toronto`

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/medikeep/uploads`	`/app/uploads`	Document uploads
`/home/jeeves/docker/medikeep/logs`	`/app/logs`	Application logs
`/home/jeeves/docker/medikeep/backups`	`/app/backups`	Backup output

medikeep-db (PostgreSQL 15.8)

Image: `postgres:15.8-alpine`

Variable	Value
`POSTGRES_DB`	`medical_records`
`POSTGRES_USER`	`medapp`
`POSTGRES_PASSWORD`	**REDACTED**

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/medikeep/postgres/data`	`/var/lib/postgresql/data`

Traefik Labels

```
traefik.http.routers.medikeep-ext.rule: Host(`medical.jeeves5454.ddns.net`)
traefik.http.routers.medikeep-ext.entrypoints: websecure
traefik.http.routers.medikeep-ext.tls.certresolver: letsencrypt
traefik.http.routers.medikeep-ext.middlewares: plex-geoblock@file,crowdsec-bouncer@file
traefik.http.routers.medikeep-ext.service: medikeep-svc

traefik.http.routers.medikeep-int.rule: Host(`medikeep.home.local`)
traefik.http.routers.medikeep-int.entrypoints: websecure
traefik.http.routers.medikeep-int.tls.certresolver: step-ca
traefik.http.routers.medikeep-int.service: medikeep-svc

traefik.http.services.medikeep-svc.loadbalancer.server.port: 8000
```

Notes / Gotchas

- `SSO_REDIRECT_URI` points to the external domain — this is the OAuth callback URL registered in Authentik. Even when accessing via the internal `home.local` URL, the OIDC callback will redirect through the external domain.
- Medikeep's `LOG_LEVEL` is set to `DEBUG` — logs may be verbose. Logs are accessible in the bind-mounted `/app/logs` directory.

- Contains sensitive personal medical data. Limit backup exposure and ensure the PostgreSQL data directory is included in regular backup schedules.

Last Updated: 2026-06-17

09-lubelogger.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: LubeLogger tags: [lubelogger, vehicles, maintenance, postgres]

Overview

LubeLogger is a vehicle maintenance and fuel log tracker. It stores service records, oil changes, tyre rotations, fuel fill-ups, and other vehicle events. External-only route with GeoBlock and CrowdSec — no internal `home.local` route. Authentication is LubeLogger's own built-in user system. Backend is PostgreSQL 16.

Access

Type	URL	Auth
External	`https://logger.jeeves5454.ddns.net`	GeoBlock + CrowdSec (own auth)

No internal route configured.

Containers

Container	Image	Role
`lubelogger`	`ghcr.io/hargata/lubelogger:latest`	Web application
`lubelogger-db`	`postgres:16`	PostgreSQL database

lubelogger (application)

Image: `ghcr.io/hargata/lubelogger:latest`
Runtime: .NET 10.0.8 (ASP.NET Core)

Key environment variables:

Variable	Value / Notes
`POSTGRES_CONNECTION`	**REDACTED** (includes password)
`ASPNETCORE_HTTP_PORTS`	`8080`
`LC_ALL` / `LANG`	`en_US.UTF-8`

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/lubelogger/data`	`/App/data`	App data and uploads
`/home/jeeves/docker/lubelogger/keys`	`/root/.aspnet/DataProtection-Keys`	ASP.NET data protection keys

lubelogger-db (PostgreSQL 16)

Image: `postgres:16`

Variable	Value
`POSTGRES_USER`	`lubelogger`
`POSTGRES_DB`	`lubelogger`
`POSTGRES_PASSWORD`	**REDACTED**

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/lubelogger/db`	`/var/lib/postgresql/data`

Traefik Labels

```
traefik.http.routers.lubelogger.rule: Host(`logger.jeeves5454.ddns.net`)
traefik.http.routers.lubelogger.entrypoints: websecure
traefik.http.routers.lubelogger.tls.certresolver: letsencrypt
traefik.http.routers.lubelogger.middlewares: plex-geoblock@file,crowdsec-
bouncer@file,lubelogger-headers
traefik.http.middlewares.lubelogger-headers.headers.customrequestheaders.X-Forwarded-Proto:
https
traefik.docker.network: traefik-net
traefik.http.services.lubelogger.loadbalancer.server.port: 8080
```

Notes / Gotchas

- `POSTGRES_CONNECTION` in the container environment contains the database password in plaintext — treat this as sensitive and avoid logging the environment of this container.
- The `DataProtection-Keys` bind mount (`/lubelogger/keys`) must persist across container restarts. If it is lost, all active user sessions will be invalidated and users will need to log in again.
- The `X-Forwarded-Proto: https` middleware is required for ASP.NET Core to correctly identify requests as HTTPS when behind a reverse proxy.

Last Updated: 2026-06-17

10-hortusfox.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: HortusFox tags: [hortusfox, plants, garden, mariadb]

Overview

HortusFox is a plant care and garden management app. It tracks plants, watering schedules, fertilisation, and overdue care tasks. A separate cron container triggers scheduled task checks. Internal-only access via Step-CA TLS — no external route. The workspace is named "Jeeves Garden".

Access

Type	URL	Auth
Internal	`https://hortusfox.home.local`	HortusFox own auth (Step-CA TLS)

No external route — LAN access only.

Containers

Container	Image	Role
`hortusfox`	`ghcr.io/danielbrendel/hortusfox-web:latest`	Web application
`hortusfox-db`	`mariadb:11`	MariaDB database
`hortusfox-cron`	`alpine:latest`	Scheduled task runner

hortusfox (application)

Runtime: PHP 8.3 on Apache

Key environment variables:

Variable	Value / Notes
`APP_WORKSPACE`	`Jeeves Garden`
`APP_TIMEZONE`	`America/Toronto`
`APP_LANG`	`en`
`APP_DEBUG`	`true`
`APP_ADMIN_EMAIL`	`jeeves5454@gmail.com`
`APP_OVERDUE_TASK_HOURS`	`10`
`APP_ENABLE_HISTORY`	`true`
`APP_HISTORY_NAME`	`Garden Log`
`APP_ENABLE_SYSTEM_MESSAGES`	`true`
`APP_ENABLE_PHOTO_SHARE`	`false`
`APP_ENABLE_CHAT`	`false`
`APP_ENABLE_SCROLLER`	`true`
`APP_CRON_PW`	**REDACTED**
`DB_HOST`	`hortusfox-db`
`DB_PORT`	`3306`
`DB_DATABASE`	`hortusfox`
`DB_USERNAME`	`hortusfox`
`DB_CHARSET`	`utf8mb4`

Bind mounts:

Host Path	Container Path	Purpose
`/home/jeeves/docker/hortusfox/images`	`/var/www/html/public/img`	Plant images
`/home/jeeves/docker/hortusfox/themes`	`/var/www/html/public/themes`	Custom themes

Host Path	Container Path	Purpose
`/home/jeeves/docker/hortusfox/logs`	`/var/www/html/app/logs`	App logs
`/home/jeeves/docker/hortusfox/migrations`	`/var/www/html/app/migrations`	DB migration files
`/home/jeeves/docker/hortusfox/backup`	`/var/www/html/public/backup`	Export/backup files

hortusfox-db (MariaDB 11)

Image: `mariadb:11` (MariaDB 11.8.6)

Variable	Value
`MYSQL_DATABASE`	`hortusfox`
`MYSQL_USER`	`hortusfox`
`MYSQL_PASSWORD`	**REDACTED**
`MYSQL_ROOT_PASSWORD`	**REDACTED**

Bind mounts:

Host Path	Container Path
`/home/jeeves/docker/hortusfox/db`	`/var/lib/mysql`

hortusfox-cron (Alpine)

Image: `alpine:latest`

A lightweight Alpine container that runs periodic cron jobs to trigger HortusFox's scheduled task processing (e.g., overdue task notifications). It communicates with the main `hortusfox` container using the `APP_CRON_PW` credential. No persistent volumes.

Traefik Labels

```
traefik.http.routers.hortusfox.rule: Host(`hortusfox.home.local`)
traefik.http.routers.hortusfox.entrypoints: websecure
traefik.http.routers.hortusfox.tls.certresolver: step-ca
traefik.http.services.hortusfox.loadbalancer.server.port: 80
```

Notes / Gotchas

- `APP_CRON_PW` authenticates the cron container's requests to the HortusFox web app. This is set as a plaintext environment variable — treat it as a secret. It is REDACTED in documentation.
- `APP_DEBUG: true` means verbose PHP error output is enabled. Disable this if HortusFox ever becomes accessible externally.
- Plant images are stored in the `images` bind mount. This directory should be included in backups to preserve photo history.

Last Updated: 2026-06-17

11-linkstack.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: LinkStack tags: [linkstack, linktree, about-me, public, website]

Overview

LinkStack is a self-hosted link-in-bio / personal landing page, serving as the public-facing "about me" page and link hub for `jeevesconsults.ca`. It responds to two separate domain names on a single Traefik route: the primary `aboutme.jeevesconsults.ca` and the root `www.jeevesconsults.ca`.

Access

URL	Auth
`https://aboutme.jeevesconsults.ca`	Public — no auth
`https://www.jeevesconsults.ca`	Public — no auth

Both hostnames resolve to the same LinkStack instance. No authentication middleware — this is a public-facing page.

Configuration

Image: `linkstackorg/linkstack:latest`

Environment Variables

Variable	Value / Notes
`HTTPS_SERVER_NAME`	`aboutme.jeevesconsults.ca`
`SERVER_ADMIN`	`ask@jeevesconsults.ca`
`PHP_MEMORY_LIMIT`	`512M`
`UPLOAD_MAX_FILESIZE`	`8M`
`TZ`	`America/Toronto`

Traefik Labels

```
traefik.http.routers.linkstack.rule: Host(`aboutme.jeevesconsults.ca`) ||
Host(`www.jeevesconsults.ca`)
traefik.http.routers.linkstack.entrypoints: websecure
traefik.http.routers.linkstack.tls.certresolver: letsencrypt
traefik.http.services.linkstack.loadbalancer.server.port: 80
```

> The `||` in the Traefik host rule matches either domain — both resolve to the same container and present the same LinkStack profile.

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/linkstack/data`	`/htdocs`	App files, config, SQLite

Notes / Gotchas

- `HTTPS_SERVER_NAME` is set to `aboutme.jeevesconsults.ca`. Even though both domains are handled by Traefik, LinkStack internally uses this value for canonical URL generation. Links may contain this hostname.

- No GeoBlock or CrowdSec on this route — it is intentionally fully public. Monitor access logs if abuse is suspected.
- LinkStack stores its own SQLite database and assets under `/htdocs`. The entire `data/` bind mount must be backed up to preserve the profile content.
- Letsencrypt will issue certificates for both `aboutme.jeevesconsults.ca` and `www.jeevesconsults.ca` via the same Traefik router — both must resolve to Centerpoint's public IP in DNS for certificate issuance to succeed.

Last Updated: 2026-06-17

12-reitti.md

kstack: book: Centerpoint Home Lab chapter: Documents & Organization page: Reitti tags: [reitti, fitness, activity-tracking, postgis, rabbitmq, redis]

Overview

Reitti is a self-hosted activity and fitness tracking platform. It records GPS tracks, check-ins, and location history and displays them on interactive maps. The platform is a 6-container stack: the core Java application, a PostGIS spatial database, Redis cache, RabbitMQ message queue, a tile proxy for map tiles, and Photon for geocoding.

External access is at `https://reitti.jeeves5454.ddns.net`. Check-in data from Foursquare/Swarm is synced into Reitti via the separate **swarm-reitti-bridge** service (see next page).

Access

Type	URL	Auth
External	`https://reitti.jeeves5454.ddns.net`	Reitti own auth + letsencrypt

No internal `home.local` route — external-only.

Containers

Container	Image	Role
`reitti-app`	`dedicatedcode/reitti:latest`	Core Java application
`reitti-postgis`	`postgis/postgis:17-3.5-alpine`	Spatial database (PostgreSQL 17 + PostGIS 3.5)
`reitti-redis`	`redis:7-alpine`	Cache and session store
`reitti-rabbitmq`	`rabbitmq:3-management-alpine`	Message queue for async jobs
`reitti-tiles`	`nginx:alpine`	Map tile caching proxy
`reitti-photon`	`rtuszik/photon-docker:1.3.0`	Local geocoding (Nominatim-based)

reitti-app

The main Spring Boot application serving the Reitti web UI and REST API. Ingests location data via the OwnTracks-compatible endpoint used by swarm-reitti-bridge.

Key environment variables:

Variable	Value / Notes
`POSTGIS_HOST`	`postgis`
`POSTGIS_PORT`	`5432`
`POSTGIS_DB`	`reittidb`
`POSTGIS_USER`	`reitti`
`POSTGIS_PASSWORD`	**REDACTED**
`REDIS_HOST`	`redis`
`REDIS_PORT`	`6379`
`RABBITMQ_HOST`	`rabbitmq`
`RABBITMQ_PORT`	`5672`
`RABBITMQ_USER`	`reitti`
`RABBITMQ_PASSWORD`	**REDACTED**
`PHOTON_BASE_URL`	`http://photon:2322`

Volumes:

Volume / Mount	Container Path	Purpose
Docker volume `reitti_reitti-data`	`/data`	Application data and uploads

Traefik labels:

```
traefik.http.routers.reitti.rule: Host(`reitti.jeeves5454.ddns.net`)
traefik.http.routers.reitti.entrypoints: websecure
traefik.http.routers.reitti.tls.certresolver: letsencrypt
traefik.http.services.reitti.loadbalancer.server.port: 8080
```

reitti-postgis

PostgreSQL 17 with PostGIS 3.5 spatial extension. Stores all GPS track data, waypoints, and geospatial indices.

Image: `postgis/postgis:17-3.5-alpine`

Variable	Value
`POSTGRES_USER`	`reitti`
`POSTGRES_DB`	`reittidb`
`POSTGRES_PASSWORD`	**REDACTED**

Volumes:

Volume	Container Path
`reitti_postgis-data` (named)	`/var/lib/postgresql/data`

reitti-redis

Redis 7 (Alpine) — used for caching and session management.

Image: `redis:7-alpine`

Volumes:

Volume	Container Path
`reitti_redis-data` (named)	`/data`

No authentication configured — network-isolated to the Reitti internal stack network.

reitti-rabbitmq

RabbitMQ 3 with management plugin. Handles async processing of imported tracks and location events.

Image: `rabbitmq:3-management-alpine`

Variable	Value
`RABBITMQ_DEFAULT_USER`	`reitti`
`RABBITMQ_DEFAULT_PASS`	**REDACTED**

Volumes:

Volume	Container Path
`reitti_rabbitmq-data` (named)	`/var/lib/rabbitmq`

RabbitMQ management UI is available internally on port 15672 (not exposed via Traefik).

reitti-tiles

NGINX-based map tile proxy/cache. Serves map tiles from upstream tile providers with local caching to reduce external requests.

Image: `nginx:alpine` (NGINX 1.29.8)

Variable	Value
`NGINX_CACHE_SIZE`	`1g`

No persistent volumes — tile cache is in-memory. No external Traefik route.

reitti-photon

Local geocoding service using the Photon engine (Nominatim/OSM-based). Configured for the `ca` (Canada) region with parallel update strategy.

Image: `rtuszik/photon-docker:1.3.0`

Variable	Value
`REGION`	`ca`
`UPDATE_STRATEGY`	`PARALLEL`

Volumes:

Volume	Container Path
`reitti_photon-data` (named)	`/photon/data`

The photon data volume holds the downloaded Nominatim extract for the CA region. It can be large (several GB); allow time for the initial data download on first start.

Data Flow

```
GPS device / OwnTracks → Reitti OwnTracks ingest API
Foursquare/Swarm check-ins → swarm-reitti-bridge → Reitti OwnTracks ingest API
                                        ↓
                                  reitti-app (Spring Boot)
                                        ↓
                                RabbitMQ (async processing)
                                        ↓
                                PostGIS (spatial storage)
                                        ↓
                            Redis (cache) + Photon (geocoding)
                                        ↓
                                  Web UI / Map tiles
```

Notes / Gotchas

- All named Docker volumes (`reitti_reitti-data`, `reitti_postgis-data`, `reitti_redis-data`, `reitti_rabbitmq-data`, `reitti_photon-data`) must be included in any backup strategy — none are bind-mounted to host paths.
- PostGIS 17 requires spatial extensions to be enabled in the database on first run — `dedicatedcode/reitti:latest` handles this automatically on startup.
- The Photon geocoding data download is region-specific (`REGION: ca`). If geocoding stops working after a restart, check that the photon data volume is intact and the photon container started successfully.
- RabbitMQ management UI (port 15672) is available inside the Docker network but is not exposed externally. Access via `docker exec` or Portainer console if needed.
- Check-in data from Foursquare/Swarm flows through **swarm-reitti-bridge** (see next page). Reitti itself has no Foursquare integration — the bridge translates and delivers data via the OwnTracks ingestion endpoint.

Last Updated: 2026-06-17

Swarm-Reitti Bridge

Overview

Swarm-Reitti Bridge (v1.1.0) is a custom-built Node.js service that automatically syncs Foursquare/Swarm check-ins into two downstream services:

- **Reitti** — self-hosted location history tracker, receiving check-ins in OwnTracks format
- **AdventureLog** — self-hosted travel journal, receiving check-ins as Locations, Visits, and World Travel region marks

It was built specifically for this homelab as a bridge between the Foursquare API and these self-hosted services — keeping all location history under local control.

The service runs at `https://swarm.jeeves5454.ddns.net` and is externally reachable so the OAuth callback from Foursquare can complete.

> **This service was built in this homelab environment.** Its source code lives at `/home/jeeves/docker/code-server/projects/swarm-reitti-bridge/` and the production deployment at `/home/jeeves/docker/swarmreitti/swarm-reitti-bridge/`. The container image (`swarm-reitti-bridge`) is built locally via `docker compose build` and is not published to any registry.

Background: Why Polling Mode?

This bridge was originally designed to use Foursquare **push notifications** — Foursquare would POST to the bridge's `/push` endpoint on every check-in, enabling near-instant syncing. That architecture worked correctly.

In **late 2024 / early 2025**, Foursquare changed their API pricing and moved push notifications behind a paid credit system. Attempting to subscribe to push notifications returns:

```
{ "errorType": "credits_exhausted", "code": 402 }
```

The bridge was updated to **polling mode** as the solution. Instead of waiting for Foursquare to push, it actively polls the Foursquare API every 15 minutes for new check-ins.

Aspect	Push (legacy, broken)	Polling (current)
Trigger	Foursquare → Bridge (instant)	Bridge → Foursquare (timer)
Latency	< 1 second	Up to 15 minutes
API credits required	Yes (paid)	No (free tier)
API calls/day	~0 idle	96 (10% of free quota)
Cost	Requires payment	$0.00

The 15-minute maximum latency is entirely acceptable for personal location history tracking.

How It Works

High-Level Flow

```
User checks in on Swarm
         ↓
Foursquare API stores check-in
         ↓
Bridge polls every 15 minutes (or immediately on container start)
GET /v2/users/self/checkins?oauth_token=...
         ↓
Bridge identifies new check-ins (deduplication via lastCheckinIds)
         ↓
┌──────────────────────────────────────────────────────┐
│  For each new check-in (oldest-first):               │
│                                                      │
│  1. Convert to OwnTracks format                      │
│     POST to Reitti ingest API                        │
│     Authorization: Bearer <REITTI_API_TOKEN>         │
│                    ↓                                 │
│             Check-in on Reitti map                   │
│                                                      │
│  2. (if AdventureLog configured)                     │
│     Find or create AL Category (Foursquare → emoji)  │
```

```
│     Find or create AL Location (dedup by name + coords) │
│     Create AL Visit (dedup: one per location per day)   │
│     Mark AL World Travel region as visited              │
│                     ↓                                   │
│          Location + Visit in AdventureLog               │
│          Region marked on world map                     │
└─────────────────────────────────────────────────────────┘

          ↓
State saved to disk (lastCheckinId, lastCheckinTimestamp)
```

AdventureLog failures are **non-fatal** — Reitti sync completes regardless.

Deduplication

The bridge tracks `lastCheckinIds` per user — a map of `userId → most recently processed check-in ID`. On each poll:

1. **Quick check** — fetches the single most recent check-in from Foursquare. If its ID matches `lastCheckinIds[userId]`, the poll exits immediately (already up to date).
2. **Gap fetch** — if different, fetches all check-ins within a 2-week window (paginated, 50 per page), sorted oldest-first so Reitti receives events in chronological order.

Startup Behaviour

On container start, the bridge immediately polls for any users already stored in `state.json` — it does not wait for the first 15-minute interval. This ensures check-ins that occurred while the container was down are processed within seconds of restart.

State Persistence

Three state objects are saved to `/app/data/state.json`:

Key	Contents
`userTokens`	OAuth access tokens per userId (from Foursquare)
`lastCheckinTimestamps`	Last seen check-in Unix timestamp per userId
`lastCheckinIds`	Last processed check-in ID per userId

Writes are atomic (write to `.tmp`, then rename) to avoid corruption on crash.

> **Permission note:** `state.json` is owned by root (written inside the container). Use `sudo` when editing on the host.

Data Mapping

Reitti (OwnTracks Format)

OwnTracks Field	Source	Notes
`_type`	Hardcoded: `"location"`	OwnTracks type identifier
`lat`	`checkin.venue.location.lat`	Venue latitude
`lon`	`checkin.venue.location.lng`	Venue longitude (`lng` → `lon`)
`tst`	`checkin.createdAt`	Unix timestamp of check-in
`tid`	First 2 chars of username, uppercased	OwnTracks tracker ID
`acc`	Hardcoded: `10`	Accuracy in metres
`alt`	Hardcoded: `0`	Altitude (not available from Foursquare)
`batt`	Hardcoded: `100`	Battery (not applicable)
`vel`	Hardcoded: `0`	Velocity (not applicable)
`t`	Hardcoded: `"c"`	OwnTracks trigger type: check-in
`desc`	`checkin.venue.name` (max 200 chars)	Venue name
`addr`	Address + City + State + Country (max 300 chars)	Formatted address string

```
POST <REITTI_API_URL>
Authorization: Bearer <REITTI_API_TOKEN>
Content-Type: application/json
```

AdventureLog Integration

Category Mapping

- The **primary** Foursquare category maps to an AdventureLog category (created if absent)

- Category icons are resolved from a static Foursquare → emoji lookup table (`adventurelog-categories.js`); unknown categories fall back to 🏷️
- **Secondary** Foursquare categories become AdventureLog `tags`

Location Deduplication

Locations are matched by `venue name (lowercase) + coordinates rounded to 2 decimal places` (≈1.1 km tolerance). If a match exists, it is reused without modification. If not, a new location is created. Coordinates sent to AdventureLog are rounded to 6 decimal places (AdventureLog enforces a 9-total-digit limit on lat/lng fields).

Visit Deduplication

One visit is created per location per local calendar day. The local date is derived using the venue's timezone (resolved from lat/lng via `geo-tz`). A check-in shout and any companion names (`with` array) are combined into the visit notes field.

World Travel (Region Marking)

After each visit is created, the bridge resolves the check-in's country and state/province to an ISO 3166-2 region code via AdventureLog's `GET /api/regions` list, then POSTs to `/api/visitedregion`. Already-visited regions are cached per poll cycle and skipped silently. A small country name map handles known Foursquare → AdventureLog naming mismatches (e.g. `México` → `Mexico`). Unresolvable regions are logged as warnings.

> **Note:** Creating Locations and Visits via the API does **not** automatically update the World Travel world map — these are separate systems in AdventureLog. The bridge handles both independently.

OAuth Setup

Foursquare OAuth 2.0 is used to grant the bridge permission to read check-in data. This is a one-time setup that persists in the state file.

Initial Setup Steps

1. **Visit the auth URL:**

```
https://swarm.jeeves5454.ddns.net/auth
```

2. **The bridge redirects to Foursquare** with the configured `client_id` and `redirect_uri`.
3. **Authenticate** with your Foursquare account and authorise the app.
4. **Foursquare redirects back** to `https://swarm.jeeves5454.ddns.net/callback` with an authorisation code.
5. **The bridge exchanges** the code for an access token, saves it to state, and begins polling immediately.

> **Important:** Re-authenticating via `/auth` resets the sync baseline to the most recent check-in at that moment — older unsynced check-ins will be skipped. Only re-authenticate if starting fresh or after a token expiry. If you need to reprocess past check-ins, patch `state.json` directly (see Operations below).

Foursquare App Configuration

Field	Value
Redirect URI	`https://swarm.jeeves5454.ddns.net/callback`
Push API URL	(not required — polling mode)

Access and Endpoints

External URL: `https://swarm.jeeves5454.ddns.net` **Certificate:** Let's Encrypt (letsencrypt resolver)
Port: 3000 (internal)

API Endpoints

Method	Path	Purpose
`GET`	`/`	Status dashboard — service cards, recent check-ins, recent errors
`GET`	`/auth`	Start Foursquare OAuth flow (rate-limited: 10/15 min)
`GET`	`/callback`	OAuth callback (Foursquare redirects here)
`GET`	`/health`	JSON health — `ok`, per-service status, last sync times
`POST`	`/push`	Legacy webhook endpoint (inactive in polling mode)

The `/health` endpoint returns structured JSON including an `ok: true/false` field suitable for Uptime Kuma's JSON Query monitor:

```
{
  "ok": true,
  "status": "healthy",
  "connectedUsers": 1,
  "reitti": { "configured": true, "lastSuccessAt": "...", "lastSuccessVenue": "..." },
  "adventurelog": { "configured": true, "lastSuccessAt": "...", "lastSuccessVenue": "..." },
  "lastPollAt": "..."
}
```

Configuration

Environment Variables

Variable	Value / Notes
`FOURSQUARE_CLIENT_ID`	**REDACTED** (Foursquare developer console)
`FOURSQUARE_CLIENT_SECRET`	**REDACTED**
`FOURSQUARE_REDIRECT_URI`	`https://swarm.jeeves5454.ddns.net/callback`
`REITTI_API_URL`	`https://reitti.jeeves5454.ddns.net/api/v1/ingest/owntracks`
`REITTI_API_TOKEN`	**REDACTED** (Reitti API bearer token)
`PUSH_SECRET`	**REDACTED** (session signing secret)
`ADVENTURELOG_API_URL`	`https://travel.jeeves5454.ddns.net`
`ADVENTURELOG_API_KEY`	**REDACTED** (AdventureLog API key)
`UPTIME_KUMA_PUSH_URL`	**REDACTED** (Uptime Kuma push monitor URL, no query params)
`POLLING_INTERVAL_MINUTES`	`15` (default)
`PORT`	`3000`
`NODE_ENV`	`production`

Omitting both `ADVENTURELOG_API_URL` and `ADVENTURELOG_API_KEY` disables AdventureLog sync entirely. Omitting `UPTIME_KUMA_PUSH_URL` disables push heartbeats.

Traefik Labels

```
traefik.http.routers.swarm-bridge.rule: Host(`swarm.jeeves5454.ddns.net`)
traefik.http.routers.swarm-bridge.entrypoints: websecure
traefik.http.routers.swarm-bridge.tls.certresolver: letsencrypt
traefik.http.services.swarm-bridge.loadbalancer.server.port: 3000
```

Volumes / Bind Mounts

Host Path	Container Path	Purpose
`/home/jeeves/docker/swarmreitti/swarm-reitti-bridge/data`	`/app/data`	State file (`state.json`) persisted across restarts

Monitoring (Uptime Kuma)

Three complementary monitors are recommended:

Monitor Type	Target	What it catches
Docker Container	Container name `swarm-reitti-bridge`	Container crash / OOM kill
HTTP(s)	`https://swarm.jeeves5454.ddns.net/health`	App unresponsive
HTTP(s) JSON Query	`https://swarm.jeeves5454.ddns.net/health` → `$.ok`	Reitti/AL sync errors, no users
Push	Uptime Kuma push monitor URL	Poll loop stopped, token expired

The push monitor receives `?status=up` after every successful poll and `?status=down&msg=<reason>` on failure. Configure the heartbeat interval to `POLLING_INTERVAL_MINUTES + 2` minutes (e.g. 17 minutes for a 15-minute poll interval).

Logging

The bridge uses structured JSON logging to stdout (Dozzle-compatible):

```
{"ts":"2026-06-23T12:00:00.000Z","level":"INFO","cat":"POLL","msg":"Scheduled
poll","userCount":1}
{"ts":"2026-06-23T12:00:01.000Z","level":"INFO","cat":"CHECKIN","msg":"Sent to
Reitti","venue":"Tim Hortons"}
{"ts":"2026-06-23T12:00:02.000Z","level":"INFO","cat":"AL","msg":"Location
created","venue":"Tim Hortons","id":"..."}
{"ts":"2026-06-23T12:00:02.000Z","level":"INFO","cat":"AL","msg":"Visit
created","locationId":"...","date":"2026-06-23"}
{"ts":"2026-06-23T12:00:02.000Z","level":"INFO","cat":"AL","msg":"Region marked as
visited","regionId":"CA-ON"}
{"ts":"2026-06-23T12:00:03.000Z","level":"INFO","cat":"STATE","msg":"State
saved","usersCount":1}
```

Log categories: `SERVER`, `POLL`, `CHECKIN`, `AL`, `AUTH`, `STATE`.

`WARN` and `ERROR` entries are also captured in an in-memory ring buffer (last 20) and displayed on the status dashboard at `/`.

View live logs:

```
docker logs -f swarm-reitti-bridge
```

Operations and Management

Verify the Bridge is Running

```
# Container status
docker ps | grep swarm-reitti-bridge

# Health check (includes ok field and per-service status)
curl -s https://swarm.jeeves5454.ddns.net/health | jq .

# Status dashboard
open https://swarm.jeeves5454.ddns.net/
```

Re-authenticate with Foursquare

If the OAuth token expires or becomes invalid (bridge logs show `Token expired` WARN):

1. Visit `https://swarm.jeeves5454.ddns.net/auth`
2. Log in and authorise the app
3. The bridge saves the new token and polls immediately

> Do **not** re-authenticate to retry past check-ins — it will move the sync baseline forward and those check-ins will be permanently skipped.

Reprocess Past Check-ins

To force the bridge to reprocess a specific check-in (e.g. after fixing an AL sync error):

```
cd /home/jeeves/docker/swarmreitti/swarm-reitti-bridge

# View current state
sudo cat data/state.json

# Roll back to just before a specific check-in Unix timestamp
sudo python3 -c "
import json
with open('data/state.json') as f:
    s = json.load(f)
s['lastCheckinIds']['<userId>'] = None
s['lastCheckinTimestamps']['<userId>'] = <tst - 1>
with open('data/state.json', 'w') as f:
    json.dump(s, f, indent=2)
print('Done')
"

docker compose restart swarm-reitti-bridge
```

The startup poll will immediately pick up the check-in on restart.

Adjust Polling Interval

Edit `POLLING_INTERVAL_MINUTES` in the compose environment and rebuild:

Interval	API calls/day	Use case
`5`	288	More responsive (30% quota)
`15`	96	**Recommended** (10% quota)
`30`	48	Most conservative (5% quota)

Rebuild the Container

```
cd /home/jeeves/docker/swarmreitti/swarm-reitti-bridge
docker compose up -d --build
docker logs swarm-reitti-bridge --follow
```

Troubleshooting

Check-ins not appearing in Reitti

1. Check the status dashboard at `/` — the Reitti card shows last push time and any error message
2. Check logs:
   ```
   docker logs swarm-reitti-bridge | grep -E '"cat":"CHECKIN"|"level":"ERROR"'
   ```
3. Verify token is connected: `curl -s https://swarm.jeeves5454.ddns.net/health | jq .connectedUsers` — if `0`, re-authenticate via `/auth`
4. Test Reitti API directly:
   ```
   curl -s -o /dev/null -w "%{http_code}" -X POST \
     https://reitti.jeeves5454.ddns.net/api/v1/ingest/owntracks \
     -H "Authorization: Bearer <token>" \
     -H "Content-Type: application/json" \
     -d '{"_type":"location","lat":43.65,"lon":-79.37,"tst":1782088106}'
   ```
   Expected: `200`. If `401`, the token in `.env` is wrong.

Check-ins not appearing in AdventureLog

1. Check the status dashboard `/` — the AdventureLog card shows last sync and any error

2. Verify AdventureLog API key is valid:

   ```
   curl -s "https://travel.jeeves5454.ddns.net/api/locations?limit=1" \
     -H "X-API-Key: <key>"
   ```

   Expected: JSON with `count` field. If `403`, the key is wrong.
3. Common error — **trailing slash causes 308 redirect with lost body**: the bridge strips trailing slashes automatically; if seeing 400s verify the AL URL has no trailing slash in `ADVENTURELOG_API_URL`

World Travel region not updating

- The bridge only marks regions when `venue.location.state` and `venue.location.country` are both present in the Foursquare data. Venues with missing location data are skipped.
- If a region resolves but the country/state name doesn't match AdventureLog's English names, a `WARN` log is emitted: `Could not resolve region`. Add a mapping to `FOURSQUARE_COUNTRY_MAP` in `index.js` if needed.
- `GET /api/visitedregion` can confirm which regions are already marked.

OAuth callback fails (redirect URI mismatch)

- Ensure `FOURSQUARE_REDIRECT_URI` in env exactly matches the redirect URI registered in the Foursquare developer console: `https://swarm.jeeves5454.ddns.net/callback`

Bridge container exits immediately

- Check logs: `docker logs swarm-reitti-bridge`
- Most common cause: missing required environment variable. The app logs all missing vars and exits with code 1.

State file corruption

```
sudo rm /home/jeeves/docker/swarmreitti/swarm-reitti-bridge/data/state.json
docker restart swarm-reitti-bridge
# Then re-authenticate via /auth
```

Source Code

```
/home/jeeves/docker/code-server/projects/swarm-reitti-bridge/   ← development
/home/jeeves/docker/swarmreitti/swarm-reitti-bridge/            ← production deployment
├── index.js                    ← Main application
├── adventurelog-categories.js  ← Foursquare category → emoji icon map
├── Dockerfile
├── docker-compose.yml
├── package.json                ← version 1.1.0
├── tests/
│   ├── integration.test.js
│   └── unit.test.js
└── data/
    └── state.json              ← Runtime state (OAuth tokens, last check-in IDs)
```

Last Updated: 2026-06-23

AdventureLog - Travel Helper

Overview

Travel Helper is a Python-based sidecar service for AdventureLog. It runs on a weekly schedule and automatically links Locations (based on their Visits) to Collections, populates day-by-day itinerary entries, and adds region-level Locations as Trip Context. It is idempotent — re-running it never creates duplicates.

It also exposes a status web page at `https://travel-helper.home.local` showing last run time, run statistics, live log output, and a manual trigger button.

Why It Exists

AdventureLog Collections support itinerary entries and linked Locations, but provides no automatic way to connect existing Visits to a Collection by date range. Without this service, every Collection must be linked manually, location by location.

Travel Helper solves this by scanning all Collections that have a `start_date` and `end_date`, finding every Location that has a Visit falling within that range, and wiring everything together automatically.

What It Does Each Run

1. **Loads** all regions, locations (with embedded visits), and collections from AdventureLog.
2. **For each Collection with a date range:**
   - Finds all Locations whose Visit dates (converted to local timezone) fall within the collection's start–end range.
   - Links those Locations to the Collection (`PATCH /api/locations/{id}`).

- Adds a day-level itinerary entry per visit date (`POST /api/itineraries`, `is_global: false`).
- Resolves the region for each matched Location (from cache → AdventureLog `region` field → Nominatim reverse geocode fallback).
- Finds or creates a region-level Location (centroid of the region).
- Links that region Location to the Collection and adds it as Trip Context (`POST /api/itineraries`, `is_global: true`).

3. **Persists** the region ↔ location ID cache to `/data/region_cache.json` so Nominatim is only called for locations not yet resolved.

Access

Type	URL
Internal	`https://travel-helper.home.local`
External	Not exposed

The status page auto-refreshes every 60 seconds when idle, every 10 seconds while a sync is running.

Configuration

Image: `travel-helper:latest` (built locally — see Maintenance)
Source: `/home/jeeves/docker/adventurelog/travel-helper/`
Bind mount: `/home/jeeves/docker/adventurelog/travel-helper/data` → `/data`

Key environment variables (set in Portainer stack environment table):

Variable	Value / Notes
`ADVENTURELOG_API_URLS`	Comma-separated, tried left-to-right. First reachable URL wins.
	`http://adventurelog-web:3000,https://travel.home.local,https://<external>`
`ADVENTURELOG_API_KEY`	Set in Portainer — never hardcode. Rotate here when key changes.
`NOMINATIM_USER_AGENT`	Must include a contact email per Nominatim usage policy.
`NOMINATIM_DELAY`	Seconds between Nominatim requests. Default `3.0`. Do not lower.

Variable	Value / Notes
`CACHE_FILE`	`/data/region_cache.json`
`CRON_DAY`	Day of week for scheduled run. Default `mon`.
`CRON_HOUR`	UTC hour for scheduled run. Default `3`.
`STATUS_PORT`	Internal port for Flask status page. Default `80`.

Networks:

- `adventurelog-internal` — direct container-to-container access to AdventureLog
- `traefik-net` — exposes the status page through Traefik

Traefik labels:

```
- "traefik.enable=true"
- "traefik.http.routers.travel-helper-int.rule=Host(`travel-helper.home.local`)"
- "traefik.http.routers.travel-helper-int.entrypoints=websecure"
- "traefik.http.routers.travel-helper-int.tls=true"
- "traefik.http.routers.travel-helper-int.tls.certresolver=step-ca"
- "traefik.http.services.travel-helper.loadbalancer.server.port=80"
```

Dependencies

Service	Role
`adventurelog-web`	AdventureLog Next.js frontend (port 3000) — primary API target
`adventurelog-server`	Django backend — reached indirectly via Next.js proxy
Nominatim (external)	Reverse geocoding for region resolution. Rate-limited at 1 req/3s
Traefik	Routes `travel-helper.home.local` to the status page

URL Resolution (Internal-First)

On each run, Travel Helper probes `ADVENTURELOG_API_URLS` in order and uses the first URL that returns HTTP 200, 201, 401, or 403. This means:

1. `http://adventurelog-web:3000` is tried first (direct, never leaves host)
2. `https://travel.home.local` is tried second (through Traefik internally)
3. The external URL is a last resort only

The resolved URL is shown on the status page. If the internal network is healthy, you should always see `http://adventurelog-web:3000` displayed there.

Nominatim Usage

Nominatim is only called when a Location's region cannot be resolved from:

1. The in-memory cache (populated from `region_cache.json` on startup)
2. The `region` field already set on the AdventureLog Location record

On first run, most locations are resolved from the existing `geocode_cache.json` (generated during the original Swarm import) without hitting Nominatim at all. Subsequent runs only call Nominatim for newly imported locations.

Requests use `accept-language=en` and `zoom=10` to return English region names at state/province level. Rate limiting: 3-second base delay with exponential backoff (3s → 12s → 48s) on HTTP 429.

Gotchas

- **No trailing slashes on AdventureLog API URLs.** AdventureLog returns `308 Permanent Redirect` for URLs ending in `/`. HTTP clients drop the POST body on 308 redirects, causing silent `400 Bad Request` failures. All endpoints in this service omit trailing slashes.
- **`adventurelog-web` listens on port 3000**, not 8000. Port 8000 is exposed by `adventurelog-server` (Django), but API calls must go through the Next.js frontend which proxies `/api/` to the backend.
- **Collections must have both `start_date` and `end_date`** to be processed. Collections with only one date or no dates are silently skipped.
- **Visit dates are converted to local timezone before matching.** A visit at 11 pm in Toronto is June 22 locally but June 23 in UTC. The visit's IANA `timezone` field is used for this conversion. If the field is absent, UTC is used as a fallback.
- **`loc_collections` is maintained in-memory across collections in one run.** This prevents overwriting a location's collection list when it matches multiple collections in the same run.
- **The `itinerary` content_type for Location is `24`.** This is an internal AdventureLog FK constant. If a future AdventureLog update changes this, the constant `LOCATION_CONTENT_TYPE = 24` in `main.py` must be updated.

- **API key expiry.** The AdventureLog API key can be rotated in AdventureLog under User Settings → API Tokens. Update it in Portainer's environment variable table for the AdventureLog stack — never in any file.

Status Page Features

Available at `https://travel-helper.home.local`:

Section	Contents
Service Status	Running/Idle badge, resolved AdventureLog URL, last/next run times
Last Run Results	Collections matched, locations linked, itinerary entries, Trip Context entries, error count
Errors — Last Run	Timestamped list of all ERROR-level log entries from the last run
Live Log	Last 80 log lines, newest first, auto-refreshes
Run Now button	Triggers an immediate sync in a background thread

`GET /health` returns a JSON summary suitable for uptime monitoring.

Maintenance

Rebuilding the image after code changes

```
docker build -t travel-helper:latest /home/jeeves/docker/adventurelog/travel-helper
```

Then redeploy the AdventureLog stack in Portainer. Portainer uses the pre-built `travel-helper:latest` image — it does not build from source.

Rotating the API key

1. Generate a new token in AdventureLog → User Settings → API Tokens
2. Update `ADVENTURELOG_API_KEY` in the AdventureLog stack's environment variable table in Portainer
3. Redeploy the stack

Clearing the region cache

Delete `/home/jeeves/docker/adventurelog/travel-helper/data/region_cache.json`. The next run will rebuild it from AdventureLog region fields and Nominatim (expect the run to take longer and make Nominatim calls).

Adjusting the schedule

Change `CRON_DAY` and `CRON_HOUR` in Portainer environment variables and redeploy. Times are UTC.

Checking logs outside the status page

```
docker logs travel-helper --tail 100 -f
```

How It Was Built

Travel Helper was developed iteratively during a session in June 2026 as a companion to the Swarm → AdventureLog bulk import (`import_swarm.py`). The original import created ~7,000 Locations and ~10,000 Visits but left Collections unlinked. Key design decisions:

- **Python + APScheduler** for the weekly cron, matching the import toolchain
- **Flask** for the status page, styled to match the Swarm-Reitti Bridge dashboard
- **Internal-first URL probing** to avoid unnecessary external traffic; the service lives on `adventurelog-internal` so direct container access is always preferred
- **Full re-scan on every run** (idempotent) rather than incremental sync, to avoid state management complexity
- **`region_cache.json` pre-seeded** from the Swarm import's `geocode_cache.json` so that ~95% of locations resolve without any Nominatim calls on first run

---

Last Updated

2026-06-24