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

Revision #1
Created 2026-06-24 05:23:29 UTC by Admin
Updated 2026-06-24 05:24:40 UTC by Admin