FLUX WIKI
STARTER KIT — ARCHITECTURE & CONTRACTS

COMPONENT CONTRACTS AN AI AGENT IMPLEMENTS — THE WORKER PIPELINE (DRAFT GENERATION) AND THE PAGE/HUB/EMBED/CATALOG BUILDERS, WITH INPUTS, OUTPUTS, AND INVARIANTS.

flux.dantesisofo.com/wiki/starter-kit-architecture/

FLUX_WIKI_v2.0

JUNE 2026

FLUX DOCUMENTATION SYSTEM Layer 11 — STARTER KIT | starter-kit-architecture
flux.dantesisofo.com/wiki/starter-kit-architecture/

STARTER KIT — ARCHITECTURE & CONTRACTS

Component-level contracts. Implement each as a function/module; verify each with the acceptance checks. Schemas referenced here are defined in CONFIGURATION.

1. DATA FLOW

```
PRIVATE/LOCAL                                              PUBLIC/STATIC (S3+CloudFront)
 street name ■canonical()■■ slug/title                       /<slug>/index.html
 originals/  ■Worker.process()■■ submission.json  ■■         /<slug>/large|thumbs/*.jpg
                                                  ■■Publisher.publish()■■ /<slug>/<slug>-project.zip
 catalog.json (index) ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■    data/catalog.json
        ■                                                    /hub/ , /hub/embed/ , /catalog/
        ■■deploy_surfaces()■■ hub + embed + listing ■■■■■■■ + CloudFront invalidation
```

2. WORKER CONTRACT — process()

Signature: process(input_folder, out_dir, title, author, location, do_geocode=True) -> submission.json

Stages (ordered; emit progress per stage):

1. **Image discovery** — list JPEGs in input_folder. Log jpegs found: N.
2. **EXIF/GPS extraction** — per photo: read EXIF, compute sha256[:16]=id, dedup by sha, build the Photo record (lat/lon/timestamp/camera fields). Silent pass.
3. **Chronological sort + keys** — sort by timestamp_unix; assign seq (001…), large=large/<id>.jpg, thumb=thumbs/<id>.jpg.
4. **Derivative generation** — write a web-sized "large" (e.g. long edge ~1600px) and a thumbnail per photo, **EXIF stripped**, into out_dir/large & /thumbs. *Slowest stage; emit* k/N.
5. **Reverse geocoding** (best-effort) — coordinates → address; **rate-limit ~1 req/s**, **cache by ~111 m grid** (round lat/lon to 3 dp). Never fatal.
6. **Stats** — compute Stats (totals, geotagged %, duration from first/last timestamp, distance summing consecutive GPS points via haversine).
7. **Route** — ordered [[lat,lon]] of geotagged photos by time → stats.route.
8. **About text** — if ANTHROPIC_API_KEY set, generate a short paragraph; else a deterministic template from stats. Set about_source.
9. **Contact sheet** — one PNG grid of all thumbnails → out_dir/contact_sheet.png.
10. **Write** submission.json — full draft per CONFIGURATION §4; set walk.collections=[COLLECTION].

Outputs: out_dir/{large/,thumbs/,contact_sheet.png,submission.json}. **Idempotent:** re-running reuses cached geocode + about (unless forced). **Acceptance:** on the sample walk, large/ and thumbs/ counts == photo count; submission.json validates; stats.route length ==

geotagged count.

3. PAGE BUILDER — `build_project_page(submission) -> html`

Self-contained static HTML for `/<slug>/`: - hero: title, author, date, one-line stats (distance in display unit); - **interactive route map** (Leaflet + `TILE_URL`; route = white casing + `ROUTE_COLOR` line); - photo grid + lightbox (uses `large`/`thumb`); - links: contact sheet, project ZIP. The page links the offline **zine** (client-side jsPDF) — no server render of PDFs. **No dependency on external CSS** beyond Leaflet; inline the rest.

4. EXPORT BUILDER — `build_project_export(run_dir, slug) -> zip`

A portable offline `<slug>-project.zip`: rewritten self-contained `index.html`, `large/`, `thumbs/`, `contact_sheet.png`, `map.geojson`, `metadata.json`, vendored `zine` assets, and base64 zine images for offline PDF. No image reprocessing.

5. CATALOG + HUB/EMBED/LISTING BUILDERS

All read **only** `catalog.json`. Pure functions of the catalog.

- `build_catalog_index(catalog) -> html` → `/catalog/` listing. **Filters OUT** walks whose `collections` contains `COLLECTION` (those live on the hub). Shows any *other* projects. (If you only run one collection, this page may be empty — fine.)
- `build_hub(catalog) -> html` → `/hub/`. **Filters IN** walks with `COLLECTION`. Renders: aggregate stats; a **coverage bar** = `min(100, round(100*n/CORRIDOR_TARGET))%`; a **master map** drawing each walk's `route` (route-only, **no markers**) with click popups (title, photo/geotag counts, distance, date, OPEN PROJECT); an archive grid + timeline.
- `build_embed(catalog) -> html` → `/hub/embed/`. Slim hub for `<iframe>`: title, stats, coverage bar, master map, ENTER button. No nav/footer/archive/timeline. Posts content height to parent for auto-resize. **Same data, same numbers as the hub.**

Map interactivity contract (hub + embed): the visible route line is `interactive:false`; a transparent **wide line (**`weight≈24, opacity:0`**)** carries click/tap/hover/popup → large mobile tap target without thickening the visible route. Popup "OPEN/ENTER PROJECT" links use `target="_top"` and are styled monochrome (override Leaflet's default blue `.leaflet-container a`).

6. CANONICAL-DISTANCE / UNIT RENDERING

Store `distance_km`; render miles (or km) at build time: `mi = km * 0.621371`. A single formatter is used by every builder so the hub and embed always agree.

7. THE SYNC INVARIANT (CRITICAL)

The hub and embed must be regenerated **together**. Implement one helper and route **all** state changes through it (see DEPLOYMENT):

```
def deploy_surfaces(s3, catalog):
    put(HUB_KEY,   build_hub(catalog))
    put(EMBED_KEY, build_embed(catalog))
    # caller also invalidates HUB_URL + EMBED_URL
```

If you ever find code uploading `HUB_KEY` without `EMBED_KEY`, that is a bug.

Next: SETUP then DEPLOYMENT.

FLUX_WIKI_v2.0 — flux.dantesisofo.com/wiki/starter-kit-architecture/