FLUX WIKI
STARTER KIT — PROJECT WORKFLOW (AGENT RUN)

RUN ONE PROJECT END TO END — NAME, UPLOAD, GENERATE, PUBLISH — VIA CLI OR PORTAL, WITH PER-STEP VERIFICATION AND THE ALIVE-VS-STUCK DIAGNOSTIC.

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

FLUX_WIKI_v2.0

JUNE 2026

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

STARTER KIT — PROJECT WORKFLOW (AGENT RUN)

Two interfaces, same pipeline. Prefer **CLI** for AI-driven runs (scriptable, verifiable); the **portal** is the human point-and-click path.

A. CLI PATH (RECOMMENDED FOR AGENTS)

```
# 1. NAME + create draft folder (canonical() derives slug/title)
python3 -m cityflux.cli new "Passyunk Ave"
#   → _drafts/passyunk-ave-in-flux/  (status: new)

# 2. UPLOAD: copy the operator's geotagged JPEGs in
cp ~/photos/passyunk/*.JPG _drafts/passyunk-ave-in-flux/originals/

# 3. GENERATE the draft (runs the Worker pipeline)
python3 -m cityflux.worker _drafts/passyunk-ave-in-flux/
#   → large/ thumbs/ contact_sheet.png submission.json  (status: draft_ready)

# 4. PUBLISH (uploads + catalog + hub/embed/listing + invalidation)
python3 -m cityflux.publisher _drafts/passyunk-ave-in-flux/
#   → live at FLUX_BASE_URL/passyunk-ave-in-flux/  (status: published)
```

Per-step verification: - after (1): `_drafts/<slug>/draft.json` exists; `slug==canonical("Passyunk Ave").slug`. - after (3): `ls large | wc -l == ls originals | wc -l`; `submission.json` validates; `stats.route` non-empty if photos are geotagged. - after (4): `curl -sI FLUX_BASE_URL/<slug>/` → 200; hub HTML contains the title; `catalog.json` has the WalkRecord.

B. PORTAL PATH (HUMAN)

`street name` → drag photos → `Generate Draft` → (review) → `Publish`. The portal calls the same Worker + Publisher in-process and shows live progress. Auth = `PORTAL_*`.

C. WHAT EACH STEP PRODUCES

Step	Writes	Status
new	_drafts/<slug>/ + draft.json	new
upload	originals/*.jpg	uploaded
generate	large/,thumbs/,contact_sheet.png ,submission.json	generating→draft_ready
publish	S3 project page+derivatives+zip, catalog entry, hub+embed+listing, invalidation	publishing→published

D. HUB AUTO-UPDATE (VERIFY THE INVARIANT)

After publish, **without any extra command**, the new project must appear on the hub AND the embed (they regenerate together — see DEPLOYMENT §5). Verify both:

```
curl -s FLUX_BASE_URL/hub/       | grep -c "PASSYUNK_AVE_IN_FLUX"   # >=1
curl -s FLUX_BASE_URL/hub/embed/ | grep -c "PHILADELPHIA COVERAGE"  # coverage bar present
```

Stats on hub == stats on embed == aggregates computed from `catalog.json`.

E. ALIVE-VS-STUCK DIAGNOSTIC (GENERATION CAN BE SLOW)

The derivative stage is the bottleneck on large originals; "slow" ≠ "stuck". To diagnose **without restarting/cancelling**: 1. **Is the count climbing?** `ls _drafts/<slug>/large | wc -l` over ~25 s — if it increases, it's alive. 2. **Instantaneous CPU** of the worker/portal process (use `top -l 2 -pid <PID>`, not `ps %cpu`, which is a misleading *lifetime average*). >0 and counts climbing = working. 3. **Status endpoint / job log** — last line names the current stage. 4. **Genuinely stuck** = counts frozen for several minutes AND instantaneous CPU ≈ 0. Only then investigate.

F. COVER IMAGE (POST-PUBLISH EDIT)

Choose any project photo as the cover → updates `catalog.cover_image` → `deploy_surfaces` re-renders hub+embed+listing → invalidation. Propagates to every card/preview. No photo re-upload.

G. EMBED ON A HOMEPAGE

```
<iframe src="FLUX_BASE_URL/hub/embed/" width="100%" height="760"
        style="border:0;" loading="lazy" title="YOUR_CITY_IN_FLUX"></iframe>
```

Stays in sync automatically as projects publish. Optional auto-resize listener: the embed posts `{height}` via `postMessage`.

Next: COMMON MODIFICATIONS.

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