

Reference

Diagrams and posters

🕒 5 min read · Updated 27 September 2026

The wiki's large process posters — what each one shows, how they are built from a declarative spec, the visual spec every poster must meet, and how to add one.

<https://docs.gurbanisoul.com/diagrams/>

The wiki has two kinds of diagram. **Mermaid** fences in any page render to accessible SVG at build time in the brand palette (on GitHub, the same fence renders with GitHub's own theme). **Posters** are large, step-by-step process pictures that the site turns into a walkthrough: press *Next* and the picture builds up one step at a time with a caption and a link to the page that explains it; *Full size* opens a zoomable, pannable view. Every poster is also a plain image on GitHub.

The posters

#	Poster	Shows	Page
01	System landscape	people, the three surfaces, the API, the pinned database, the sources, and CI as the only path to production	Architecture overview
02	Three repositories and the pins that join them	data → platform → app hand-offs by lock file, one version number	Three repositories and pins
03	Anatomy of a line record	a PDF page → units → the 23 columns of the <code>lines</code> table; what the FTS index reads and its bm25 weights	Anatomy of a line record
04	Corpus pipeline and gates	<code>rebuild_all.sh</code> stage by stage: extraction, reconcile, golden suite, atomic installs, the database layers, the integrity gate, CI on every pull request	Corpus pipeline and gates
05	The search waterfall	<code>do_search</code> as coded: normalise, mixed script, explicit modes, the twelve auto-mode tiers, the ranking	Search waterfall
06	The fold in three places	the five steps of <code>roman_norm</code> , its three homes, the 24,719 golden vectors	The Roman fold
07	The verification engine	normalise, candidates, scoring, the verdict ladder with the code's thresholds, the Ang modifier	Verification engine
08	Request lifecycle	<code>GET /api/ang/712</code> from the browser to SQLite and back, in code order	Request lifecycle
09	Bounded contexts and the gateway	five contexts, module slicing, routing generated from the code, the proofs	Bounded contexts and the gateway
10	The CI gates map	every workflow and job on a pull request, what the rulesets require, the deploy chains, the schedules	CI gates

#	Poster	Shows	Page
11	The delivery pipeline	branch → PR → staging → release PR → production → tag → the app follows; rollback; the watchers	Branching & delivery flow
12	iOS release and integrity	vendor sync, one number; the archive's gates, the ledger, the launch-integrity and bookmarks ladders	Database pair and launch integrity
13	The structure of the Granth	Angs 1–1430 as one bar; the opening, the thirty-one raags, the closing sections, the Vaars, the voices — every number from the pinned database	The structure

All thirteen posters of the plan are in place.

How a poster is made

A poster is **generated from a small declarative spec**, not drawn by hand: docs-site/posters/NN-slug.mjs lists the nodes (with a kind, a position and text), the edges, the groups and the steps; the poster kit turns that into the SVG and the steps sidecar under docs/diagrams/posters/. The kit (docs-site/posters/kit.mjs) draws on a 1200-px canvas for the ~630 px the page gives a poster on a 1280-px screen, so its smallest text shows at 13 px or more, and it routes every line at right angles around the boxes. The generated files are committed so GitHub and the docs_check gate can read them; CI fails if a committed poster differs from its spec (node scripts/build-posters.mjs --check).

```
cd docs-site
node scripts/build-posters.mjs          # (re)generate every poster
node scripts/build-posters.mjs --check  # what CI runs
```

Node kinds and what they mean — the legend of every poster uses the same words:

Kind	Looks	Means
box	cream, gold border	a component in this project
store	kraft cylinder	data at rest
actor	maroon pill	a person
gate	red block	a gate: fails the build or the deploy
pin	dashed grey box	a pin: a reviewed lock file
ext	blue-bordered box	an external system
good	green block	a proven, verified state
note	plain box with a gold bar	an explanation
fail	red-bordered box	an outcome that stops here

The legend also names the two line styles it uses: solid for a flow of data or control, dashed for “reads, pins or derives from”.

The spec every poster meets

Enforced by `tools/docs_check.py` on every pull request:

- `viewBox="0 0 1200 H"` and `data-kit="2"` (built by the kit), `width="100%"`, no fixed height; `role="img"` with a `<title>` (the alt text) and a `<desc>` of at least 40 characters.
- Colours only from `docs/brand/tokens.json`, as CSS custom properties with brand fallbacks (correct as a plain image, dark-themed when inlined). Brand red never appears; gold is a fill, never text. Text is at least 25 px (13 px as shown); Source Serif 4 for the title, the system sans for labels. Every text-on-fill pair is at least 4.5:1 in both themes, and text on a saturated fill is white by day and ink by night.
- A title block, a legend, a version stamp (`vX.Y.Z · verified YYYY-MM-DD · <sha7>`) and a footer `Source of truth: <files>` naming files that exist.
- Every step is a `<g id="step-NN" data-step>` group, and the `.steps.json` sidecar lists the same ids with a title, a caption and a link.
- **No scripture.** Posters carry technical tokens only (a raag's name, a section's name — never a verse).
- Every poster is referenced by at least one page, with alt text.

Adding a poster

The steps, and what kit 2 checks before it will build a poster, are on [Adding a poster](#).