HOW WE BUILD

The Team of **One.**

How a single founder and an in-the-loop AI agent design, ship, document, and operate an entire multi-surface product — and the operating system that makes it repeatable.

A Reliqra engineering field note · Dane DeValcourt & Claude

Reliqra is an iOS app, an Android app, a browser extension, a cloud backend, a marketing site, an admin console, and a public status page. Behind them sit a billing system, a cross-marketplace reconciliation pipeline, AI-assisted identification, and dealer-grade pricing data. It is designed, built, and operated by one person.

That sentence usually invites a single follow-up: **how?** The honest answer is that the work is not done by one person typing faster. It is done by one person who has turned the practices a healthy engineering organization relies on — version-control discipline, code review, CI/CD, documentation, observability — into a system that an AI coding agent can execute, session after session, without losing the plot.

None of the individual practices are novel. What's interesting is what happens when you treat them not as team rituals but as an **operating system for a single human and an agent** — and write that system down so precisely that it runs the same way every time.

01 Leverage over headcount

The defining constraint isn't budget or skill — it's that there is no team to hold context in their heads. So context doesn't live in heads. It lives in the repository, in a form an agent re-reads at the start of every session. The human keeps the three things that delegate poorly: **direction, taste, and the final merge.** Everything in between — decomposition, drafting, testing, documenting, deploying — is execution the agent can own.

> The agent doesn't remember last week. The repository does — and that turns out to be the more durable kind of memory.

02 The constitution

Every session boots from a set of standing instructions in a `.claude` directory — the closest thing the project has to a constitution. It encodes the decisions that must never be re-litigated: who owns what, how the brand renders, how Git is handled, what "done" means. Four layers do the work.

THE OPERATING SYSTEM

- **Instructions** — canonical, non-negotiable rules loaded into every session: entity & brand canon, Git hygiene, the definition of done.
- **Memory** — atomic facts, one per file, behind a recall index, so a fresh agent starts knowing what the last one learned.
- **Skills** — repeatable procedures the agent invokes by name: branded-PDF generation, the design system, end-of-session wrap-up, parity audits.
- **Hooks** — deterministic guardrails the harness enforces, not the model: surfacing the toolbox on startup, nudging when docs drift, refusing an unsafe command before it runs.

03 One source of truth

Planning and execution are split on purpose. A project tracker is the command center — one initiative, one project per launch gate, from app submission through go-live. **GitHub is where the work actually happens**: every change is an issue and a pull request against a protected `main`, reviewed before it merges. Even when the reviewer is, in effect, the founder plus an agent, the shape holds — nothing reaches production unseen.

The discipline compounds. Because project state lives in structured tools rather than memory, it can be **read back automatically**: a launch dashboard regenerates every thirty minutes from the tracker, the issue counts, and live health probes — a self-updating answer to "how close are we?" that no one maintains by hand.

ANATOMY OF A SESSION

Branch › Change › Pull request › Review › **Merge** › Deploy › Docs › Status › Wrap-up

Every unit of work runs the same loop. The last step records what was learned, so the next session starts smarter.

04 Documentation that writes itself

Most documentation rots because keeping it current is someone's separate job, and that someone is busy. Here the rule is inverted: **documentation is push, not pull.** Any session that changes how the system behaves updates the matching document in the same session — part of the definition of done, not a follow-up ticket.

The library is built so an agent can navigate it: a consistent taxonomy, machine-readable front-matter, and **gap tags** dropped inline wherever a fact still needs a decision or verification. Those tags surface on a dashboard and get burned down automatically; only the ones that truly need the founder's call wait for the founder.

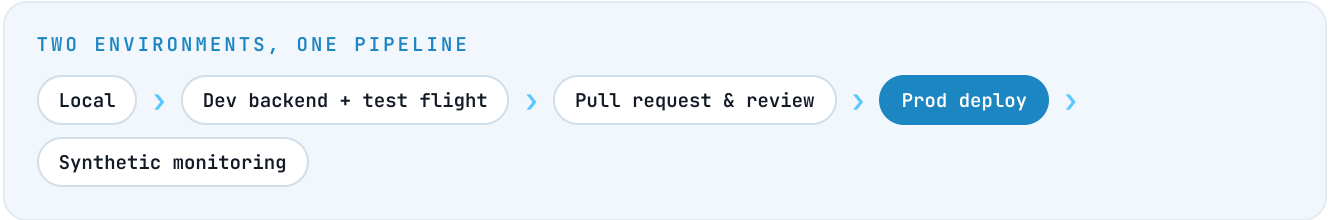
On every stable change, the whole library rebuilds itself into a browsable site and a print-ready PDF. The documentation stays true not through willpower but because **staying true is wired into the workflow** — the agent that changes the system is the one that records the change.

It is, in a real sense, self-generating: the instructions tell the agent to document, the agent documents, and the result is checked in for the next agent to build on.

> Docs don't drift when changing the system and describing it are the same act.

05 Ship it safely

DevOps for a team of one is still DevOps. **Dev and prod are real, separate environments** — a development backend and a developer test track for the apps, with production behind its own domains and protections. A change is exercised in dev before it is near a customer. Infrastructure is scripted, not remembered: backend stacks, one-command deploys, edge-function promotion, and CDN invalidation are codified, so shipping is a reproducible step in the loop — and the script is its own documentation.



Observability is held to the same honesty standard as everything else. The public status page does not ping for a polite `200` and call it green; it runs **synthetic checks that actually sign in, read the database, fetch live pricing, and call the API** — the things a customer does. If a real capability breaks, the page turns red and alerts fire. Monitoring that can quietly lie is worse than no monitoring at all.

The human in the loop

What's notable here isn't that an agent writes code; that part is increasingly ordinary. It's the **shape of the collaboration.** The human sets direction, makes the irreversible calls, applies taste, and approves the merge. The agent decomposes the work, drafts and tests it, deploys it, documents it, and — crucially — writes down what it learned, so the next session begins further ahead than the last one ended.

That last property is what turns a clever tool into an operating system. Each session leaves the repository a little more capable of running itself: one more skill, one more guardrail, one more decision recorded so it never has to be made twice. **The work compounds.**

The team is one. The leverage is not — and the difference between them is almost entirely written down, in plain text, where the next agent will read it.

> One person sets the direction. A disciplined system — and an agent that reads it — does the rest.

COLOPHON

Produced with Reliqra's own in-repo design system and branded-report tooling — the same workflow described in these pages. The wordmark, palette, and layout come from the shared system, and the document is generated from source and verified rather than laid out by hand.