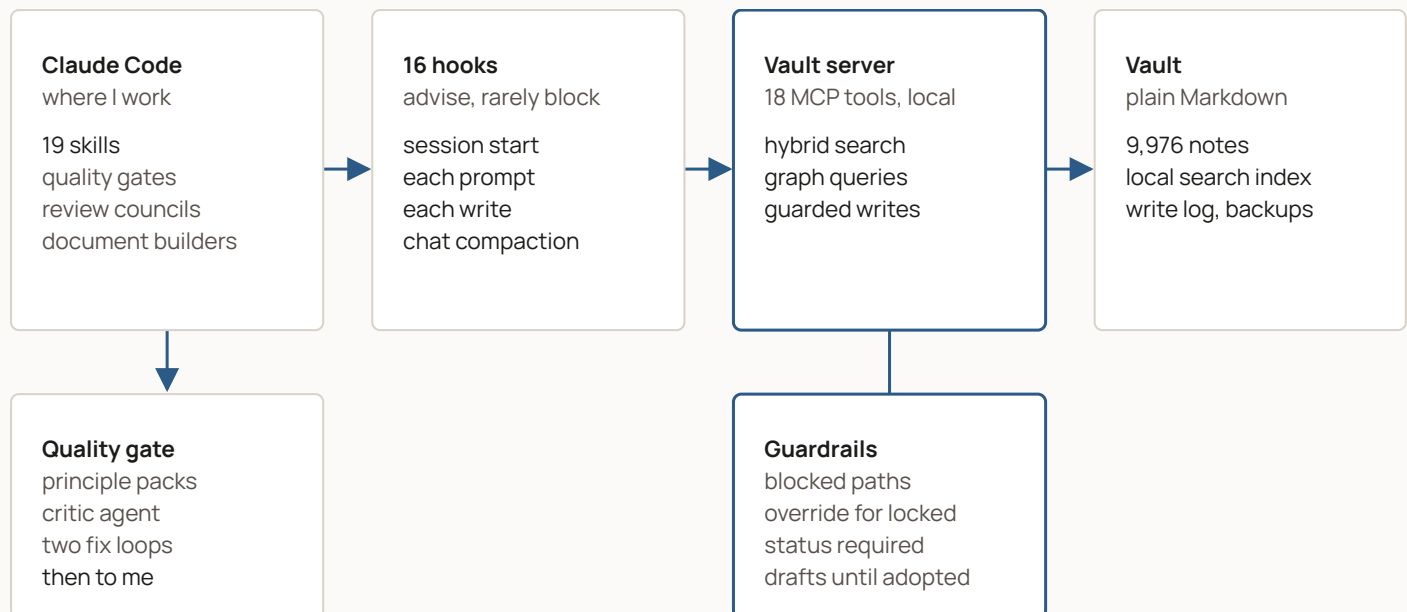


My AI operating layer

I let AI work on my own knowledge every day through one interface I built. In its vault server I decide, in code, what the model may change, and documents, pages and decks are scored before I see them.

How a request moves through it



Six decisions behind it

1. The model reaches my notes through typed tools over the Model Context Protocol. Every client works on the same vault, and the model reads notes through tools, so nothing has to be pasted into a chat.
2. Search is hybrid: local embeddings and keyword scores over heading-sized chunks, merged by rank, so search can return a note whose idea matches when its wording does not.
3. Write rules live in the server, so writes made through it follow them whatever a prompt says. Configuration folders and the rules file are blocked, locked notes need an explicit override, and a note without a status is refused.
4. Every write through the server lands in a daily log, and every patch backs up the old version first.
5. Agent notes in my project folders stay drafts until I adopt them, so I decide what becomes a record.
6. Documents, pages and decks are scored before I see them, against principle packs built from Norman, Galitz and Filipiuk for design and from my own writing rules. A critic agent flags any critical rule it breaks; after two fix loops, anything still open ships with the issues listed.

Where it stops

- One user on one Mac. The server talks to Claude Code over a local pipe (stdio) with no authentication, which is acceptable on a single-user laptop and unsafe on a network.
- The write rules cover the server only. Claude Code can still edit files directly, and my machine-safety rules are written instructions that nothing enforces.
- My prompts, and the note excerpts the model reads, go to Anthropic because the model runs there. The search index stays on the laptop.
- Hooks print hints that I can ignore, and answer quality has no formal evaluation yet.

Build a small one in four weeks

By my estimate each step is a weekend's work if you are comfortable with Python. By the end of week one it answers questions from your own notes.

Prerequisites

A folder of Markdown notes (Obsidian is optional), Claude Code, Python with the official MCP SDK, Hugging Face transformers with the BAAI/bge-small-en-v1.5 model, rank_bm25 for keyword search, and pytest. Everything except Claude runs on your laptop, and the open-source parts are free.

Week	Build	Skill you learn	The team version
1	A read-only MCP server with two tools, search and read. Register it with Claude Code and ask questions about your notes.	How MCP works, and how to describe a tool so the model uses it well.	A server over your decision records and standards, so assistants answer from your own decisions.
2	Local hybrid search. Split notes by heading, embed the chunks on your laptop, add keyword search and merge the two rankings.	Embeddings and retrieval, and how chunking changes the answers you get.	Answers that cite the exact section of your own documents they came from.
3	Guarded writes. Block your configuration folders, require a status field, back up before every edit and log every write.	Least privilege and audit trails for agents that can change things.	Agent output that stays a draft until a named person adopts it.
4	A session-start hook that loads the notes you need within a 3-second budget, and one skill that scores output against your own standards.	Context engineering, and how to check model output.	A quality gate built from your architecture principles, and a review council of agent reviewers that check a design against them.

Guardrails to set on day one

- Keep the server on stdio on your own machine, and add authentication before anything listens on a network.
- Keep secrets and client data out of the vault.
- Log every write from the first day, so the first unexplained change has a history.

Sessions start with my context loaded

- The notes relevant to what I am working on are surfaced when a session starts, so I re-explain context less often.
- Long reading goes to helper agents that return a short answer with sources, which keeps my own conversation small.

The weeks map to the Claude Certified Architect exam

They cover four of its domains: MCP, prompting and context engineering, evaluation and testing, and governance and safety. I passed the Professional exam in August 2026.

The architecture in full

Everything runs on my Mac except the model. Blue is the path a request takes and the server that enforces the rules; the dashed line is where data leaves the laptop.

