

# How my vault dreams

*My notes go stale the moment I learn something new. So every night a Mac Mini rereads my day, quietly fixes every older note it makes out of date, and leaves me an Undo button with breakfast.*

Freddie Chambers · October 2026 · a walk-through of the design

Written as the finished system. The build is in progress - the safety groundwork is done and the rest is landing one checked step at a time.

- |                                          |                                     |
|------------------------------------------|-------------------------------------|
| 01 The idea in one minute                | 07 Never overwriting my phone       |
| 02 The whole system on one page          | 08 The night shift                  |
| 03 One voice note, step by step          | 09 The 7am briefing and Undo        |
| 04 Every line says how sure it is        | 10 Why Undo, not an approval queue  |
| 05 The rule checker                      | 11 How it was built                 |
| 06 The ledger: one diary for every agent | 12 What to steal for your own setup |

01

## The idea in one minute

Picture a librarian who reads everything you learnt today, then walks the shelves fixing every book it just made out of date. That is the whole thing.

I have an Obsidian vault of around five thousand notes: people, projects, plans, meeting notes, articles. Recording a call already gives me a lovely new meeting note (my voice-note pipeline does that). The problem is everything else. The five older notes that call just made wrong carry on saying the old thing, until I fix them by hand. Which I never do.

Dreaming closes that gap. Straight after every recording, and again at 2am, the Mac Mini works out which existing notes the new information touches and adds small, labelled, quoted lines to them. It is the brain sorting the day while I sleep, hence the name. The idea comes from Andrej Karpathy's sketch of a wiki an AI keeps up to

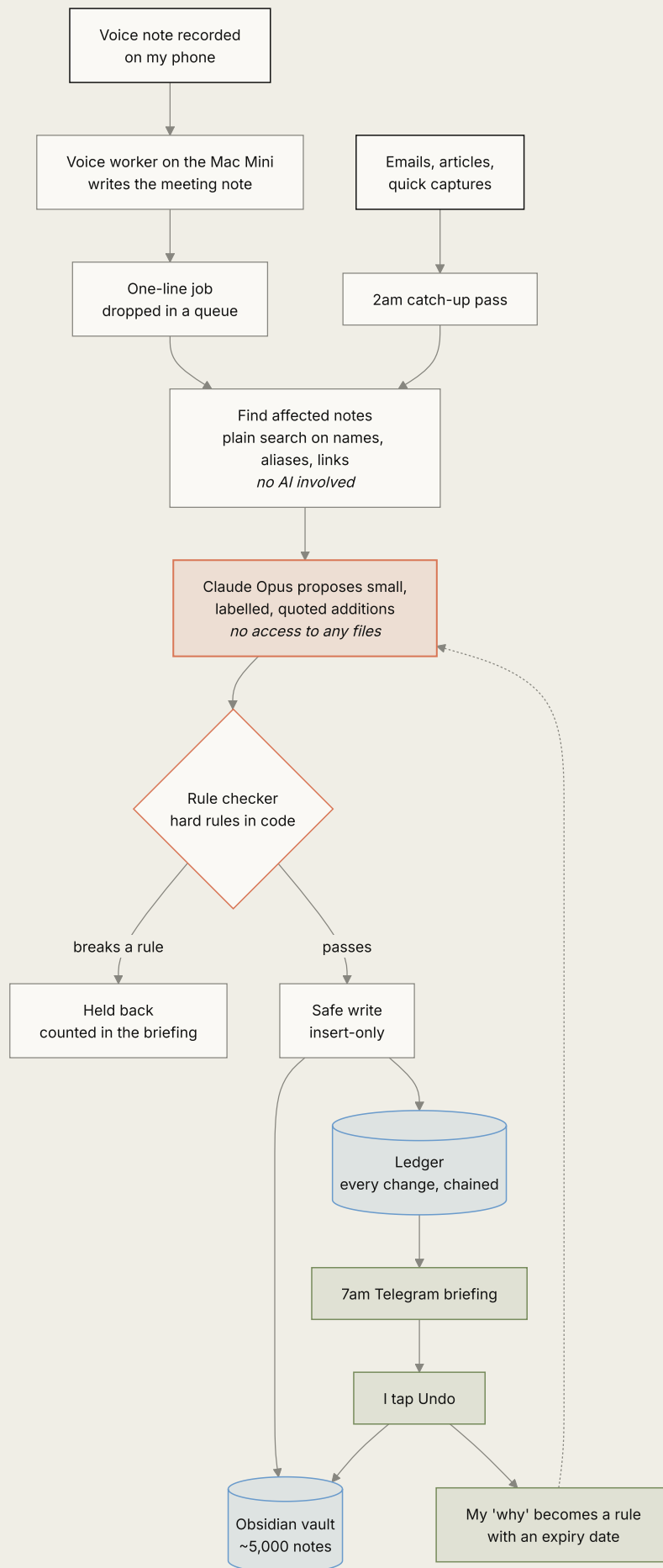
date for you, and a Code with Claude workshop on agents that dream over their own memories.

The design bet in one sentence: let it act on its own from day one, and make every single change visible, sourced and undoable in one tap - rather than making me approve each edit.

02

## The whole system on one page

Four moving parts. A **trigger** (a new voice note, or the 2am catch-up). A **dream** that finds affected notes and proposes edits. A **rule checker** written in plain code that refuses anything unsafe. And a **ledger** that records every change so the morning briefing can offer Undo.



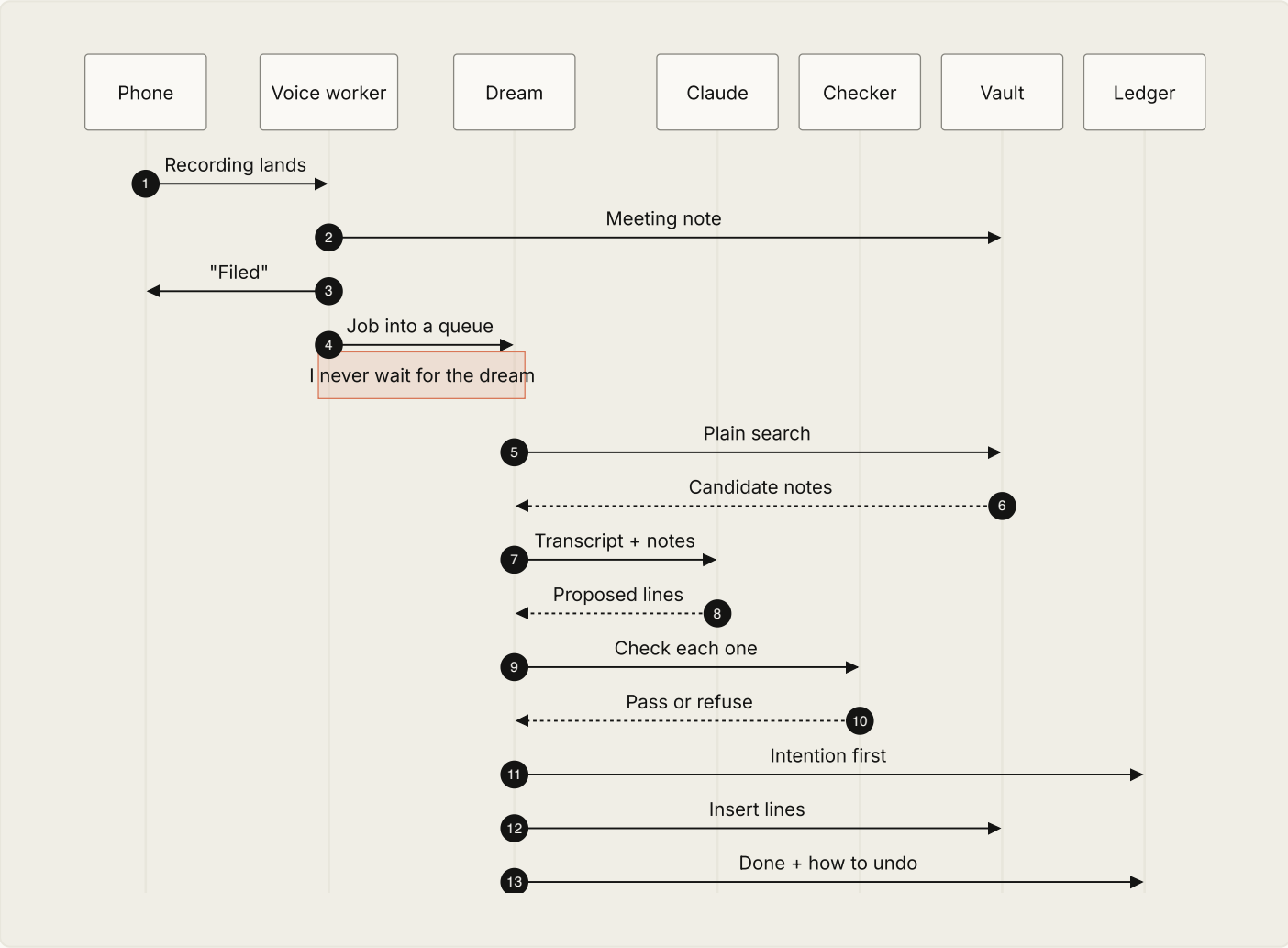
Coral = the AI step. Blue = stored state. Green = where I come in. Everything else is ordinary, testable code.

The important shape: the model never touches a file. Claude only ever returns a structured list of proposed lines. Finding notes is plain search, checking is plain code, writing is plain code. The AI is one box in the middle, boxed in on both sides.

03

## One voice note, step by step

Say I have a call about a project, and someone mentions the launch has slipped to November. Here is what happens in the few minutes after I stop recording.



A few minutes later the project hub, the person card and the plan each gain one labelled line about the November slip, all listed in the ledger.

Three details that matter

- **Queue, not inline.** The voice worker drops a job and moves on. If the dream is slow or broken, my meeting note still arrives on time.
- **Search is dumb on purpose.** Candidate notes come from matching names, aliases and links already in the vault. It is predictable, cheap and easy to test, and the model cannot wander off and decide to rewrite something unrelated.
- **New people or projects become questions, never new notes.** If the call mentions someone the vault has never heard of, the dream asks me about them in the morning rather than inventing a card.

04

Every line says how sure it is

Every machine-written line carries a label, a link to the exact source, a timestamp and the exact quoted words. So when I read a note in six months I can tell at a glance what I wrote, what the machine heard, and what the machine guessed.

| LABEL       | WHAT IT MEANS                           | NEEDS A QUOTE?                                                  |
|-------------|-----------------------------------------|-----------------------------------------------------------------|
| observed    | Someone actually said or wrote this     | Yes, word for word, from the raw source                         |
| inferred    | Worked out from something that was said | Yes, the line it was worked out from                            |
| speculative | A reasonable guess, flagged as one      | Points at the source, no quote needed                           |
| conflict    | Two sources disagree                    | Both sides, recorded side by side, never settled by the machine |

What it looks like inside a project note (examples made up for this page):

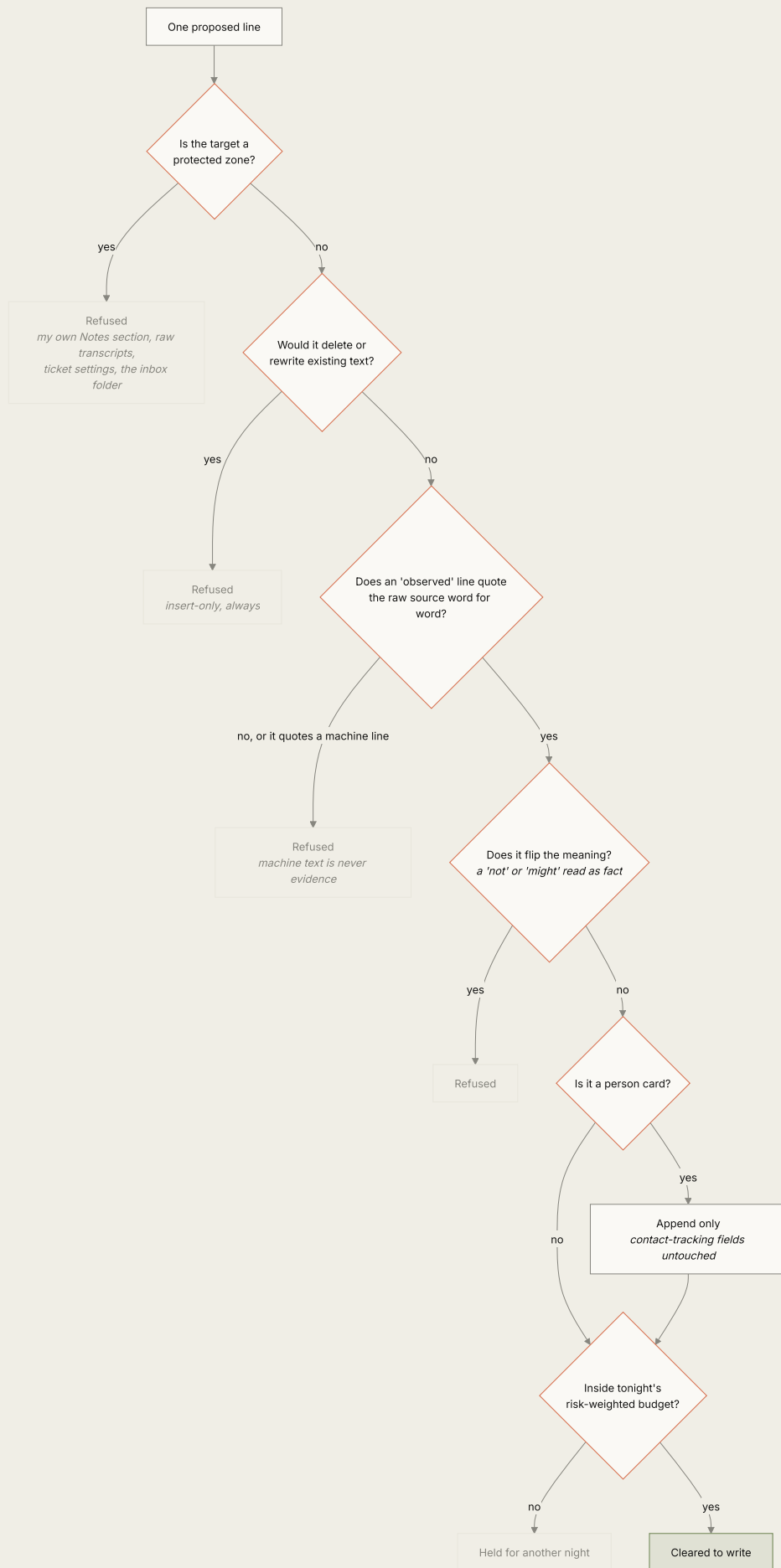
```
## Timeline
- Launch planned for October ← my line
- Launch now expected in November (observed - [[Call - Product sync (30-09-26)])
14:22 - "we're looking at November now") %%dream L000047%%
- Installer pilot likely slips too (inferred - [[Call - Product sync (30-09-26)])
14:31 - "if the installers slip we lose October") %%dream L000050%%
- (conflict - this note says 93% of homes; [[Call - Data review (02-10-26)]) 09:05 -
  "it's about thirteen percent of homes") %%dream L000048%%
```

The %%dream L000047%% tag at the end is an Obsidian comment, invisible in reading view. It is the line's ID in the ledger, which is how Undo finds exactly this line and nothing else.

05

## The rule checker

Claude proposes, code disposes. Every proposed line runs this gauntlet before anything touches the vault. None of it is a prompt asking the model to behave - these are hard checks in Python, with tests that try to sneak bad edits past them.



Refused and held lines are not thrown away silently. The briefing tells me how many were blocked, and why.

## **The snowball problem, and the one rule that stops it**

The danger with any self-updating memory is the snowball. The machine writes a plausible guess on Monday, reads it back on Tuesday as if it were a fact, builds on it on Wednesday, and by Friday a rumour has become gospel inside your own head.

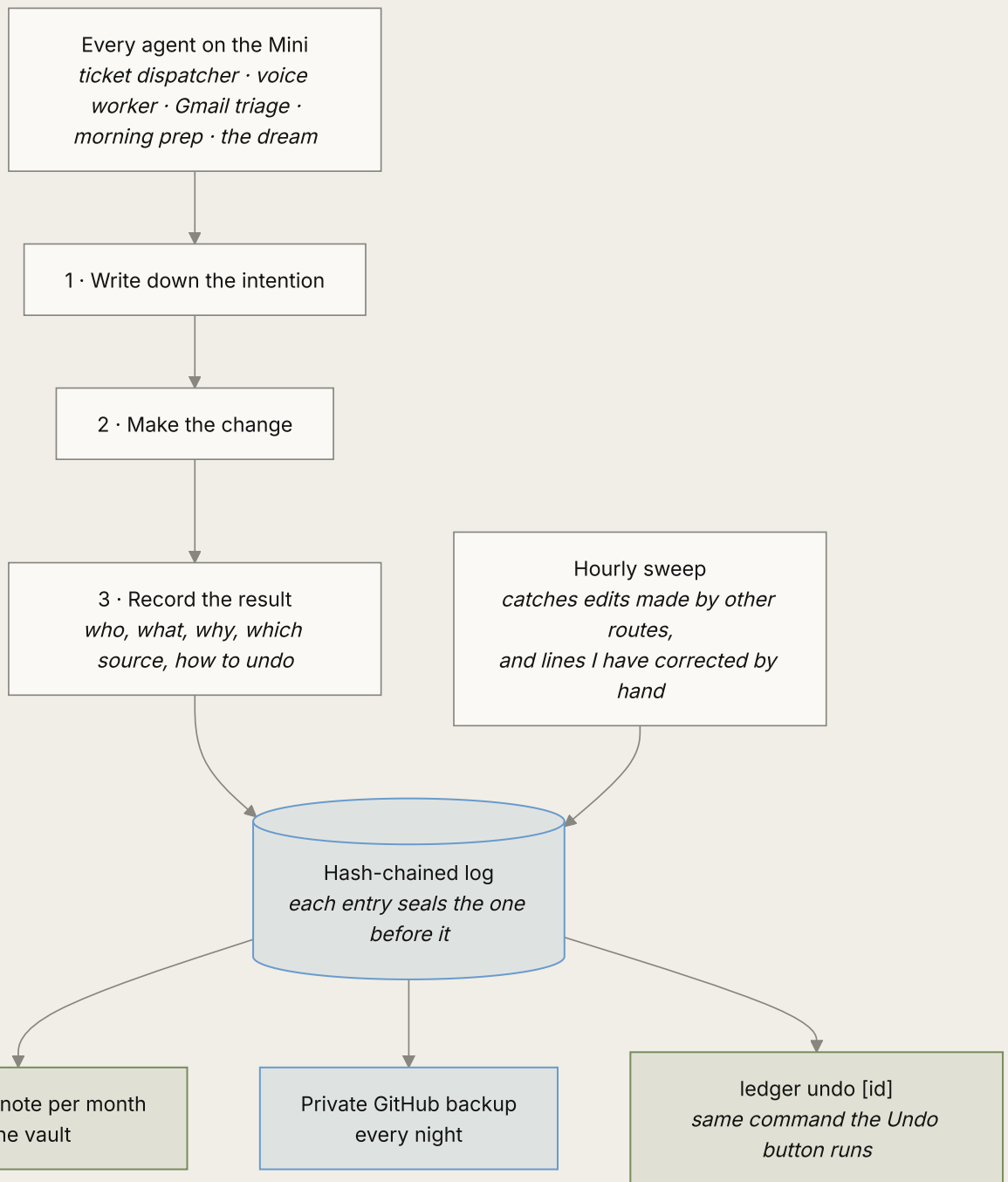
Machine text is never evidence. An "observed" line must quote a raw source: the transcript, the email, the article. Never another machine-written line. So a guess can never promote itself into a fact by being repeated.

And because every line remembers its source, corrections cascade backwards too. If a transcript turns out to have misheard a speaker, every line built on that segment is rewritten or withdrawn on its own.

06

## **The ledger: one diary for every agent**

Dreaming is not the only thing writing to my vault. A ticket dispatcher, the voice-note worker, a Gmail triage job and the morning prep jobs all edit notes too. So the dream came with a bigger idea: one locked diary of every change any agent makes, which every agent reads before it edits.



The log lives on the Mini outside iCloud, in its own git repository, with a copy of the old note text stored for every change.

---

### Intention first

Every agent writes down what it is about to do before it does it. If the Mini crashes halfway through an edit, the ledger still knows, and recovery can finish or reverse it. There is never an edit with no record.

### Tamper-evident

Each entry contains a fingerprint (hash) of the one before. Quietly editing, deleting or cutting off the end of the history breaks the chain, and `ledger verify` says exactly where.

---

### My edits count

If I fix a machine line by hand, the hourly sweep notices, and it counts exactly like tapping Undo. Correcting the vault the normal way is feedback too.

---

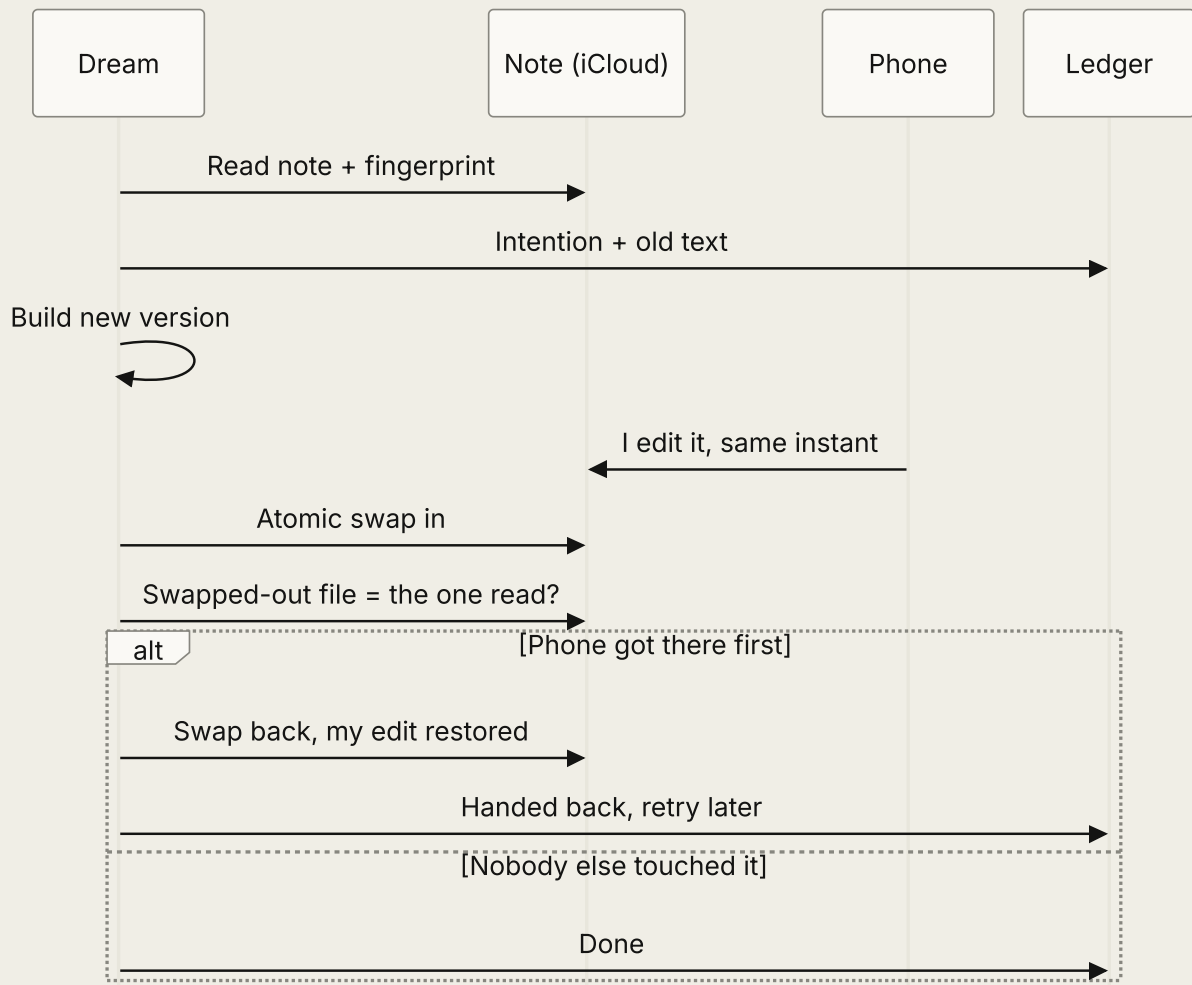
### Honest about gaps

Edits made by other programs between sweeps are only seen as hourly snapshots. The ledger records them as exactly that, rather than claiming coverage it does not have.

07

## Never overwriting my phone

The vault syncs through iCloud, so I might be editing a note on my phone at the exact moment the Mini writes to it. The classic "read the file, check it, then replace it" has a gap of a few milliseconds where my edit can vanish. We found that gap with a test, then closed it with an atomic swap on macOS: swap the new version in, then check that what got swapped out is exactly what was read. If not, swap straight back.

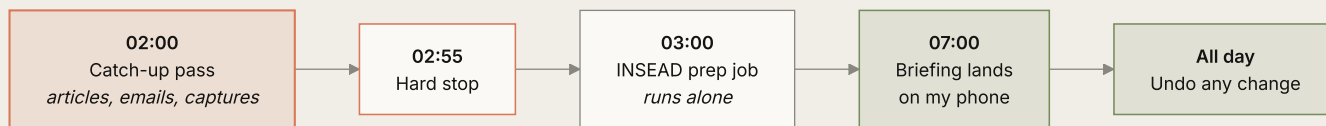


Every agent writes through this one gate, not just the dream. Insert-only, heading structure frozen, intention first, recoverable after a crash.

08

## The night shift

Voice notes are handled within minutes. Everything slower - newsletters, emails, articles, quick captures - waits for the 2am pass, which also looks for one pattern worth telling me about. It must be backed by two independent sources, and some nights there is none.



The dream always stops at 2:55 so it never collides with the 3am prep job.

## Budgets, not vibes

- Its own allowance. The dream runs on a separate Claude subscription from the one my ticket dispatcher uses, so it can never eat the dispatcher's allowance.
- Books the cost before it spends it. Each model call is costed up front against a weekly limit. If the dispatcher shows signs of trouble, the dream stands down.
- Risk-weighted edit budget. A line on a project hub is cheap. A line on a person card costs more. A busy night spends the budget on the safest edits first.

09

## The 7am briefing and Undo

Every morning a short Telegram message, 150 to 250 words, in plain English. What changed, one Undo button per note, at most one thing worth noticing, the questions it could not settle, one edit shown with its full reasoning so I can spot-check it, how many edits it held back, and this week's undo rate.

DREAM · 07:00

Good morning. 6 notes updated overnight, 2 held back.

---

Product launch hub: launch now  
November (observed)

Undo

---

Installer pilot plan: likely slips too  
(inferred)

Undo

---

Market sizing note: conflict on % of  
homes

Undo

---

+ 3 more

Show all

**Worth noticing:** two separate calls this week  
mention installer capacity as the bottleneck.

**Question:** who is "Sam from the supplier"? No  
card found.

Undo rate this week: 4%. Run healthy.



The life of one machine edit.

### Undo teaches it

- Tapping Undo removes the machine's change within a minute, even while the dream is mid-run, and asks why.
- My few words become a rule in a file the next run reads, with an expiry date so old rules do not pile up forever.
- If I keep undoing one kind of edit, it stops making that kind on its own and tells me.

- If I stop reading the briefing, it offers to switch to weekly rather than shouting into the void.

10

# Why Undo, not an approval queue

Every reviewer I showed this to suggested starting read-only, or with an approve-each-edit queue. I deliberately did not. An approval queue is just another inbox of suggestions to click through, and the whole point is that I never do the admin. So the trust comes from making autonomy safe to reverse, not from asking permission.

|                        | APPROVAL QUEUE                     | AUTONOMY WITH UNDO                           |
|------------------------|------------------------------------|----------------------------------------------|
| Notes stay current     | Only when I get round to approving | Every morning, automatically                 |
| My effort              | Review every edit                  | Skim, tap Undo on the odd one                |
| Safety net             | Me, before                         | Rules in code before, ledger and Undo after  |
| Learning               | None, unless I write rules         | Every Undo becomes a rule                    |
| Failure it watches for | Bad edits                          | Quiet drift, tracked by the weekly undo rate |

## Tested before switch-on, then trusted

Before it went live it replayed my last 20 voice notes against a throwaway copy of the vault, and I skimmed the report. That was a test before switch-on, not a trial period. After that, it acts on its own.

## What would make me change course

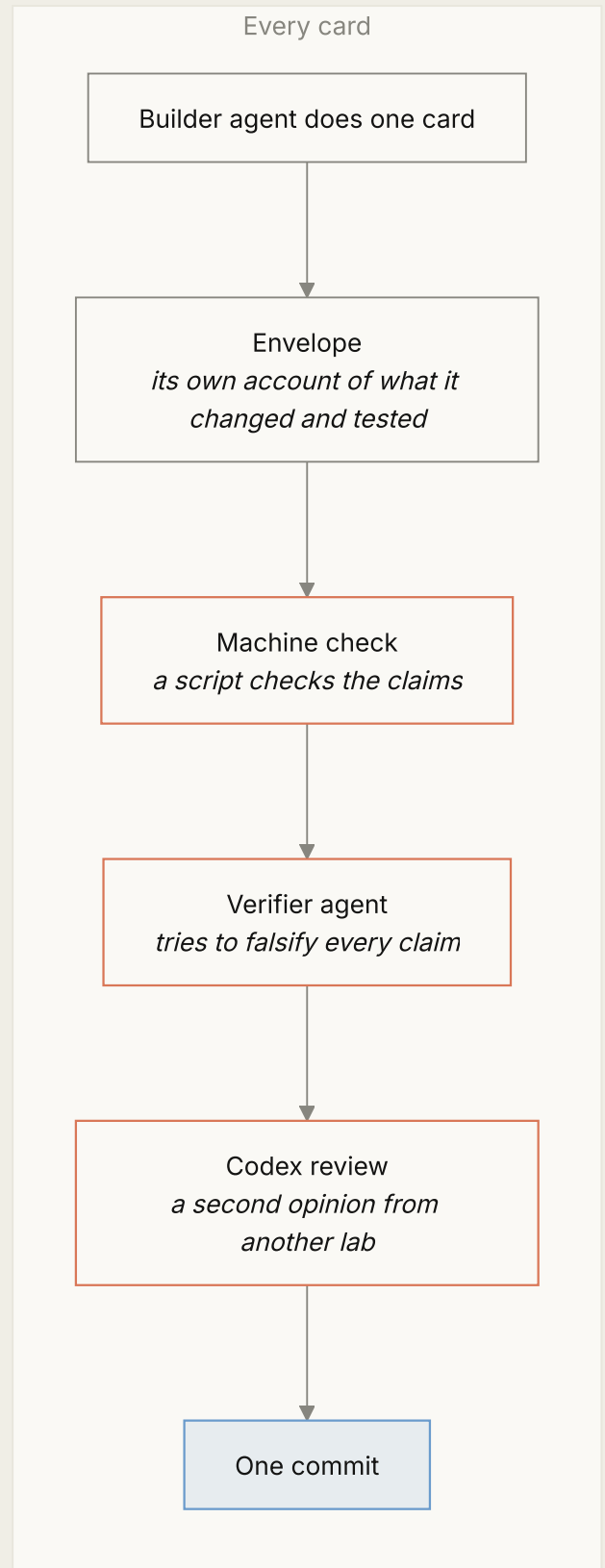
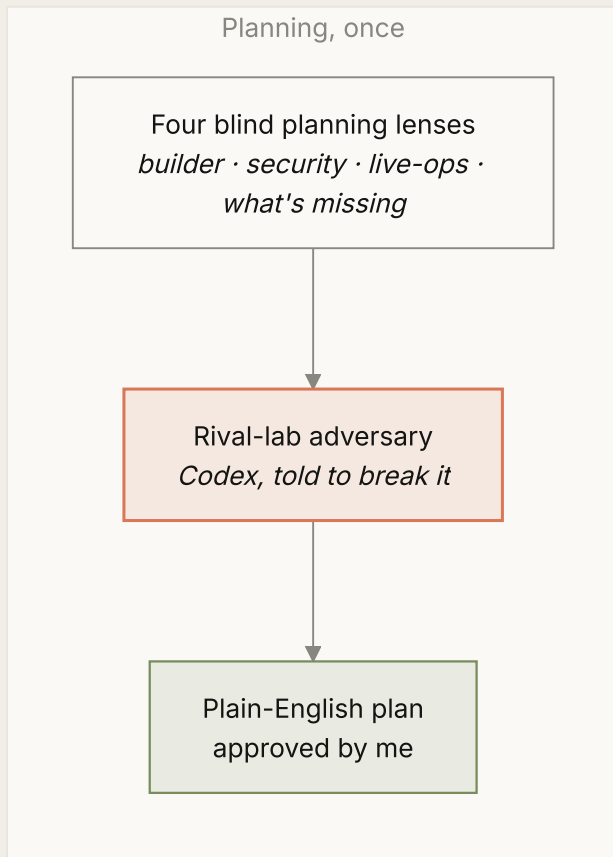
The plan named the facts it rests on in advance, each with a pre-agreed response if it turns out false:

- If it ever picks the wrong note more than about one time in seven, it narrows to hubs and projects only. No people cards, and still no approval step.
- If it ever turns a quote's meaning around, "observed" lines are restricted to near word-for-word restatements.
- If an edit of mine is ever lost despite the safe write, the dream writes only into its own separate notes, never into mine.
- If Claude cannot reliably return the structured proposal format, it moves to the API's structured output.

11

## How it was built

Built with Claude Code using my "crew" method: a hardened plan, then one small task card at a time, each one attacked before it is kept. 44 cards across four phases: safety groundwork, the ledger, the dream, the briefing.



Planning on the left, the gate every card passes on the right. Hard stops - anything live, paid or that sends a message - wait for my yes.

## Things the gates caught

- The few-millisecond race where a phone edit could vanish, found by a test and fixed with the atomic swap in section 07.
- A ledger whose tests passed but missed tampering. A randomised fuzzer that generates thousands of mangled histories now proves the chain catches it.
- A rule checker that read section boundaries one way while Obsidian read them another. It now reads every note four ways and protects to the latest possible end.
- A fuzz test that could not actually fail. Lesson learnt: prove the test catches a known-bad version before trusting a green.

12

## What to steal for your own setup

1. **Keep the model away from the files.** Let Claude propose structured edits and have ordinary code check and write them. You can test code. You cannot unit-test a vibe.
2. **Make machine text inadmissible.** Anything the AI wrote can be read, but never cited as evidence. That one rule kills the snowball.
3. **One ledger for all your agents.** Intention first, result after, hash-chained. It turns "something changed my notes" into "this agent, this reason, undo with one command".
4. **Insert-only.** An agent that can only add labelled lines cannot destroy anything you wrote.
5. **Undo beats approval.** If reversing is cheap and visible, you can let agents act. Track the undo rate as your drift alarm.
6. **Protect your own words.** Fence off the sections you write yourself, in code, not in a prompt.

Stack: Python on a Mac Mini · Claude Code headless ( `claude -p` ) with Opus for proposals · Obsidian over iCloud · a git-backed, hash-chained ledger with a private GitHub mirror · a Telegram bot for the briefing and Undo.

