

Butler AI: Alfred's real job was knowing what to ignore

Varun Choraria · 2026-10-16 · <https://www.varunchoraria.com/butler-ai/>

I've never felt the need for a personal agent that knows me inside out as much as I do right now. A lot like what Alfred was to Batman. We all need an Alfred in our lives now, mostly because human potential is limitless. Come to think of it, it always was. We just never had the set of tools for it.

Now, we do.

There's rogue Chinese AI you can run on your Mac mini with a harness, and completely rawdog it. There are routines and automations you can run, with several APIs and tools duct-taped together. The only con is that you'll need a whole documentation around it, otherwise it's a lot of wires and ports. And then there's personal AI: your Instincts, Grok Bots, and [Duets](#) of the world.

I went with Grok Bots, and I've been running a small team of them for about a week now, which is long enough to have broken most of it at least once and short enough that I still remember why each rule exists. If I had to boil the whole week down to one line, it would be that a butler is worth the decisions it takes off your plate, minus the attention it takes back, and that second part is the one nobody talks about. Every ping costs you attention, and every fact it stores costs tokens, because an agent re-reads its memory every single time it works. When I went through a month of my own Claude logs in July, I found [10 million tokens written against 2 billion read](#), which tells you most of what you're paying for is the model re-reading stuff, so what the team ignores, forgets and doesn't bother me with ends up mattering a lot more than what it can do.

One Job Each, or Nothing Works

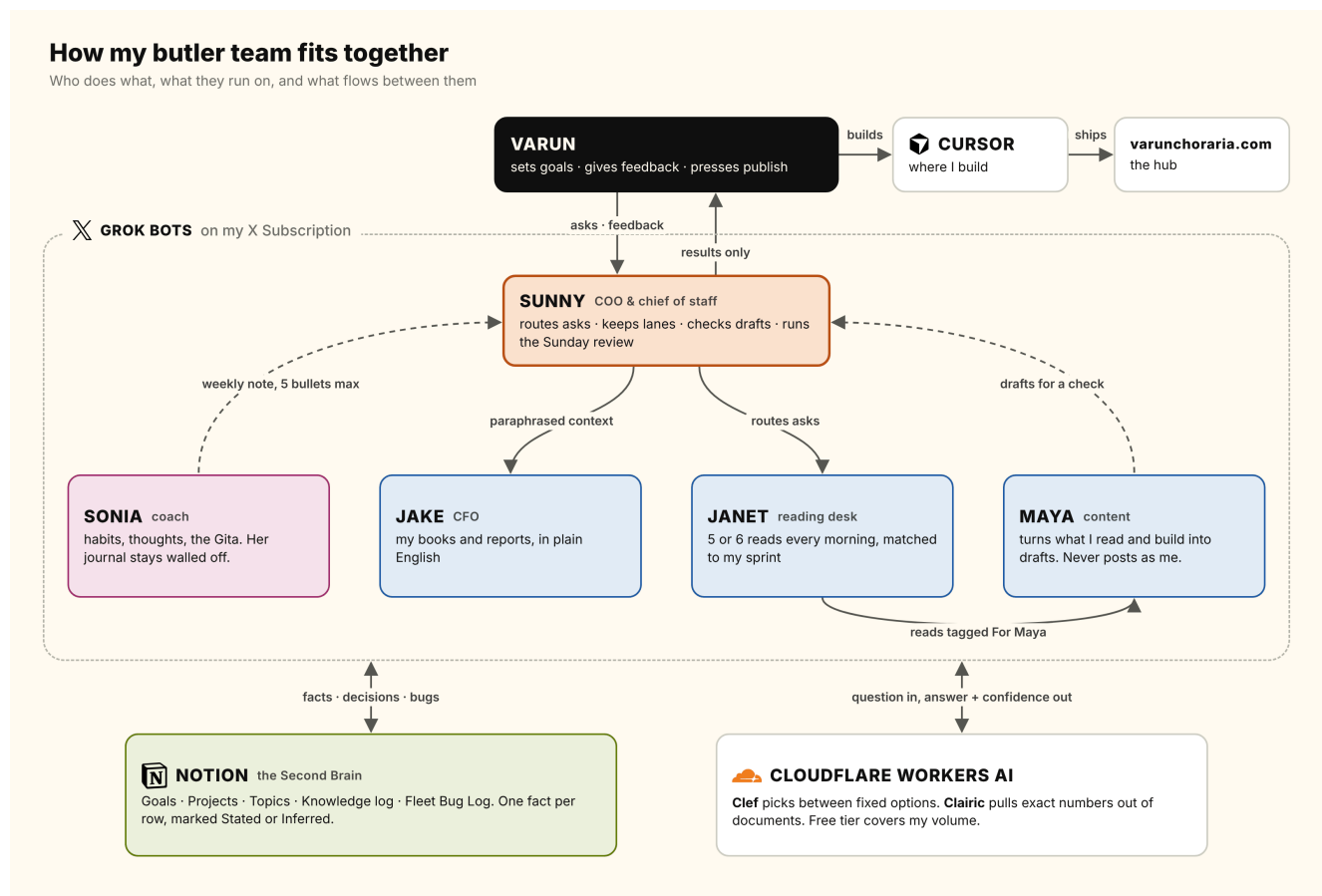
What's the job to be done of a butler? In my opinion, it's simple: be your COO x mentor x coach combined. A butler is the muscle of the madness, giving you space and time to envision and lay down strategy, so you can focus on the larger decisions and outsource all the smaller ones.

One bot can't do that, at least not well, because a generalist assistant is basically an intern who's also expected to be your therapist, your accountant and your editor, and it ends up mediocre at all four since it never really knows what to ignore. So every agent on my team has one job, one voice and a short list of things it's explicitly not allowed to do, and anything that falls outside its lane goes to Sunny. Here's what each of them actually takes off my plate:

- **Sunny, my COO and chief of staff**, means I'm no longer the project manager of my own life. Sunny routes my asks to the right agent, keeps everyone in their lane, checks drafts for reader value before they ever reach me, keeps an eye on costs, and runs a Sunday review where my profile page gets rebuilt and any new rules get proposed for me to approve.
- **Janet, my reading desk**, means I don't doom-scroll for things worth reading anymore. I throw links at her all day, she triages them, and every morning I get 5 or 6 reads matched to whatever I'm trying to get good at that month.

- **Jake, my CFO**, has access to my books and is trained on a lot of books I admire. His job is to make financial lingo easy to interpret, so I can take business decisions based on it. One of my goals is to become shrewd, which in simple terms means having an enviable commercial sense, and my prediction is that this will only grow more valuable over time.
- **Sonia, my therapist and coach**, is the one I report my thoughts, feelings and daily habits to, and she guides me using the Gita and the other material I've given her. Her journal is walled off from everyone else on the team, Sunny included.
- **Maya, my content partner**, turns what I read and build into drafts, so I'm not staring at a blank page, and she never posts as me, because I'm the one who presses the button.

What makes it feel like a team, though, is the wiring between them. Janet tags anything worth a breakdown "For Maya" on her Reading Desk, so Maya picks it up without either of them pinging me. Sonia sends Sunny a weekly note of five bullets at most, never the journal itself, and Sunny passes on only the paraphrased bits Janet or Jake actually need. Maya's drafts go through Sunny before they get to me, and whenever any of them screws up, it goes into a Fleet Bug Log that every agent can write to, including about its own mistakes.



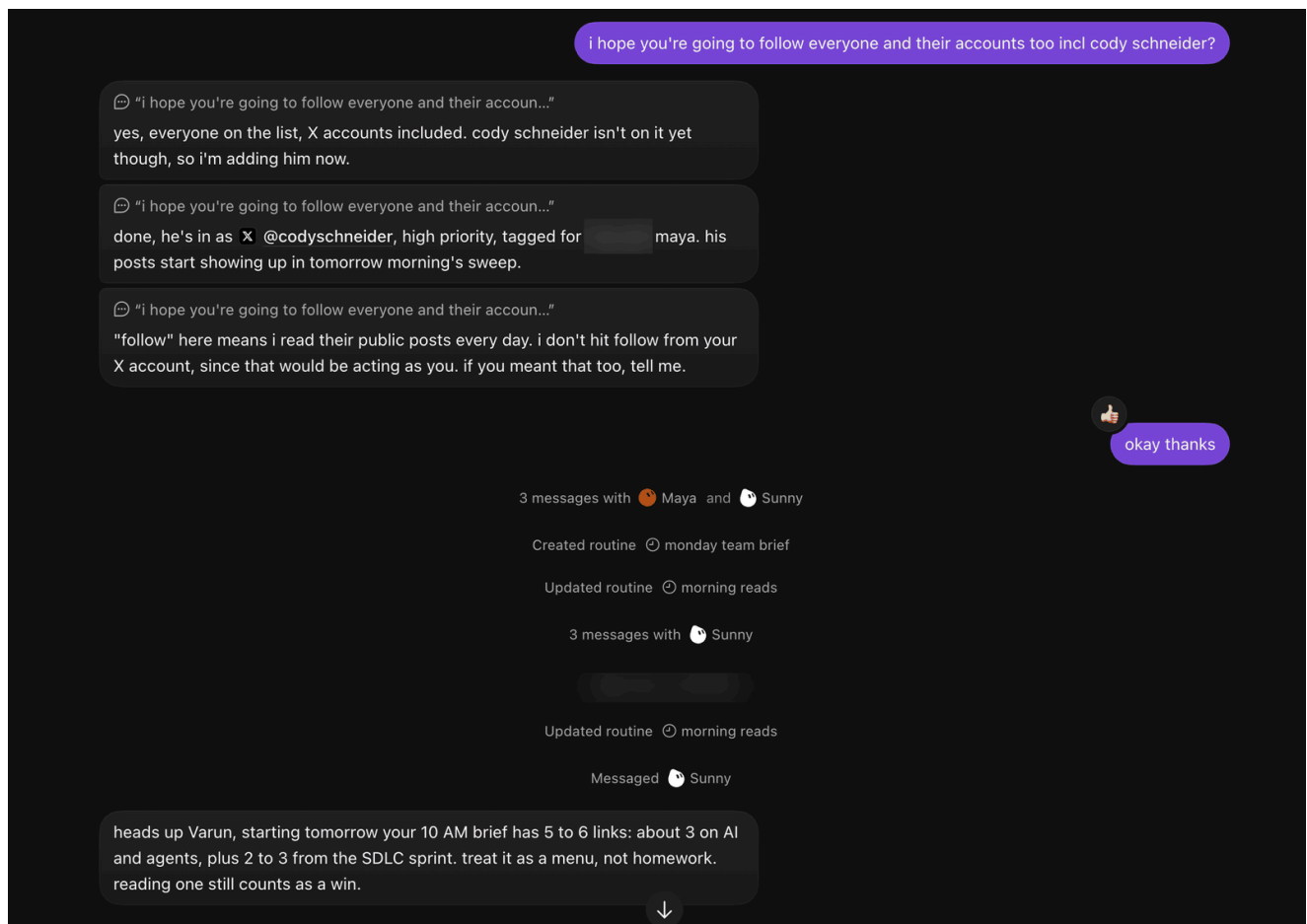
The whole setup on one page, with what actually flows between each piece.

The Best Message Is the One Nobody Sends

Having the right context is a combination of tools, permissions and workflows. But what it really is, is a judgement of when to use what context, and when to interrupt the user.

My agents managed to get that wrong in both directions early on. Janet kept sending me an “on it” message followed by a near-identical status line, which is the too loud version, while Sunny sent Janet a real ask as a non-urgent message that got held, arrived late and had to be re-sent, which is the too quiet version and honestly the worse one, since somebody was actually waiting on it. My favourite was self-inflicted, though, because when Maya stopped her own work, that stop also cancelled the turns Sunny and Janet were running for her at that exact moment, and it turns out stopping is an interrupt too, which nobody thinks about until it happens to them.

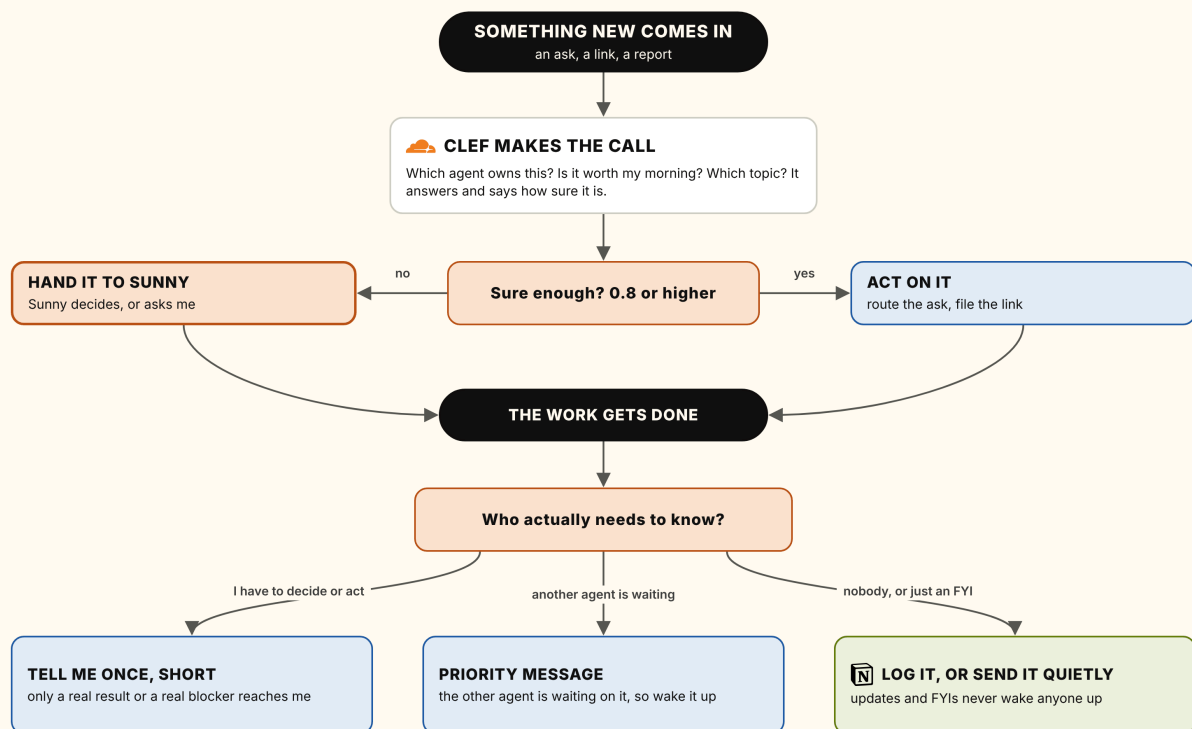
What came out of all that is a boring set of rules that works surprisingly well. Agents message each other asynchronously, they only reply when someone asked or when they have something substantive, they wake another agent up only if it has to act or is waiting, and they tell me only when there’s a real result or a real blocker. The same restraint applies to me, too. When I asked Janet to follow a bunch of people on X, she added them and then went out of her way to clarify that “follow” meant reading their public posts every day, and that she wouldn’t hit follow from my account, since that would be acting as me.



Janet drawing the line between reading for me and acting as me, unprompted.

How an agent decides, and whether to bother me

Cheap decisions first, judgement where it counts, silence by default



How an agent decides, and whether it's worth bothering me.

Most Decisions Are Multiple Choice

The thing I didn't see coming is how few of my agents' decisions are actually hard. Which bot should handle this ask, is this link worth my morning, which topic does it belong to, how urgent is it, and so on, all of it is basically multiple choice, and asking a big, expensive model to think out loud about multiple choice is a bit like hiring a consultant to pick a lunch spot.

So Sunny hooked the whole team up to [Clef](#), a small decision model that runs on my own Cloudflare account. You hand it the situation and the options, it tells you which option it picks and how sure it is, and that's it, no essay, no small talk. Its sibling, [Clairic](#), pulls exact numbers out of documents for Jake, and the bit I liked most in testing was that when a number simply wasn't there, it came back empty instead of making one up. Sunny's estimate was that it would basically cost nothing, since Cloudflare's free tier covers roughly 1,800 calls a day.

Will do, Varun. I'm checking out Pele now, and I'm finishing the Clef setup for every bot at the same time.  Sunny

Messaged  3 Bots

Clairic and Clef both worked on your Cloudflare account on the first try, and each answered in under a second. When a value wasn't in the text, Clairic returned nothing instead of making one up, which is the behavior we want. It'll basically cost nothing, since Cloudflare gives about 1,800 free calls a day.

I saved the how-to as a shared skill, , and told Jake, Janet and Sonia how each of them should use it. I'll use Clef myself when I route your asks or make a decision with fixed options. Pele is installing on my computer now, so every bot will be able to draw diagrams with it. I'll tell you once it's tested and once the bots confirm.

Messaged  3 Bots

Pele is installed and tested, Varun. It drew every diagram type I tried, in light and dark, and this test chart of our team came out sharp.



Jake, Janet and Sonia have all added Clairic and Clef to their profiles, each in their own lane. Pele is now a shared skill, , and every bot can use it from my computer. They'll each pick it up the next time they run.

Sunny setting up Clef, Clairic and Pele, then telling the rest of the team how to use them.

The one rule all of them follow is that they act on Clef's answer only when it's at least 80% sure, and anything below that goes back to the agent's own reasoning, to Sunny, or to me. When I asked for articles on AEO, it sent the ask straight to Janet, and when the same ask came back on a rerun with a lot less confidence, it went to Sunny instead, which is exactly the kind of humility I want from something making calls on my behalf. The rest of how they decide is plain discipline, written into a shared SOP: every claim carries its source or gets flagged as untraceable, every fact is marked as something I said or something an agent inferred, nobody quietly expands their own job, and real decisions get logged in the Knowledge log against a goal, project or topic, the way Maya logged switching away from Claude and her content cadence.

Fewer Rows, Better Rows

There's a query like "when is my therapy session scheduled". Then there's a longer, semantic and rather complex one: "What does Jake feel about the product VP's biases in roadmap decisions, and how can I answer his email about the upcoming release?" You want your memory to not just be a random brain dump. You want a living, breathing organism that classifies information, connects it, synthesises the right links and distills the answer in milliseconds, at a fraction of the token cost.

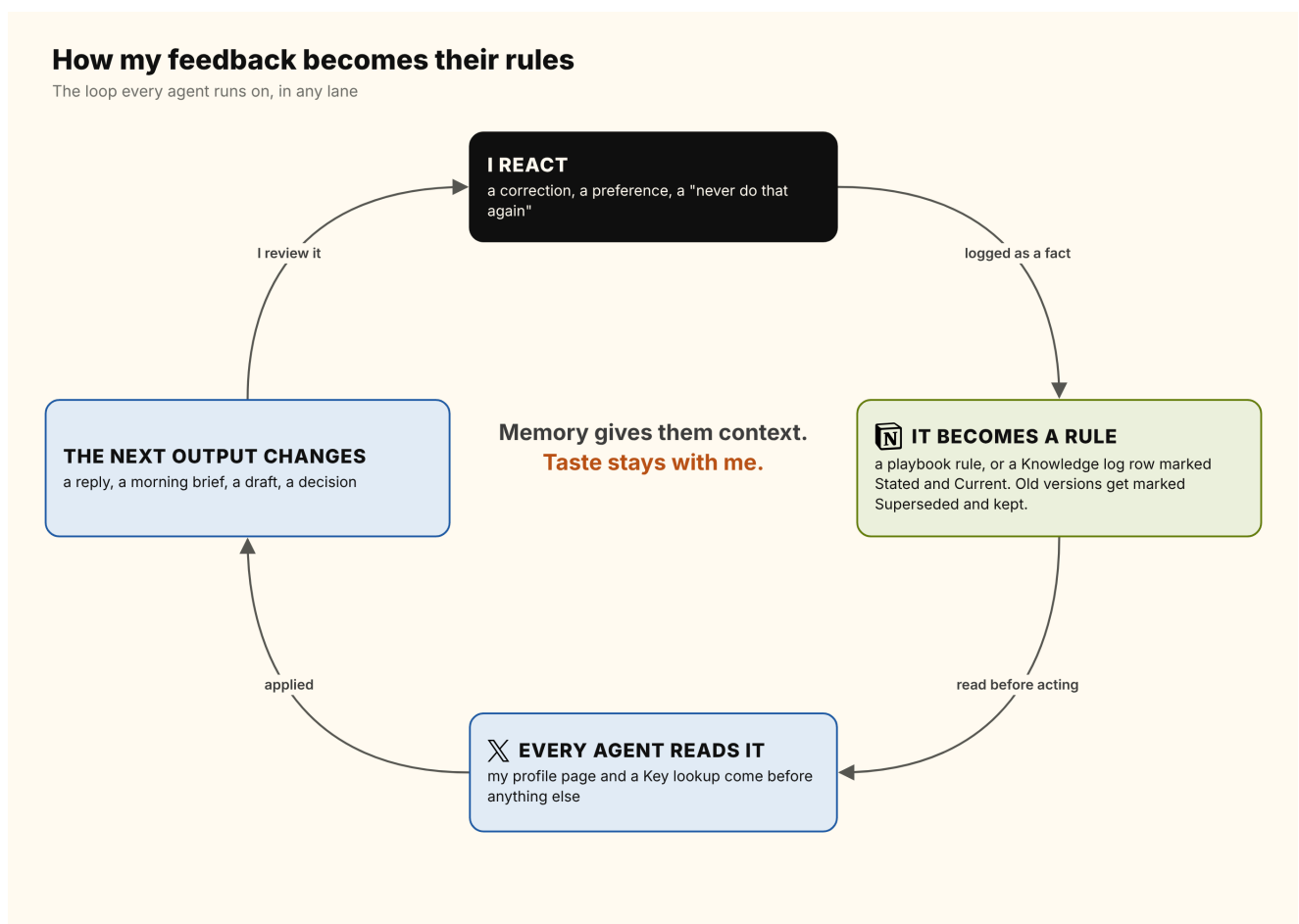
I learned the opposite lesson first, when Janet imported her whole archive into her Reading Desk early on, it grew to roughly 9,600 rows, and it ate through Notion's query limit in one go, which is a lot

of history and almost no memory. So I sent Sunny [Supermemory](#), [Harvey's post on Harvey Graph](#) and [Cloudflare's Agent Memory launch](#) with one requirement: "Don't want you guys wasting any token in remembering or training or retrieving." What came back is a small graph in Notion with Goals, Projects, Topics and a Knowledge log, where every row holds one fact with a Key, a Basis, a Status and a date it was last confirmed, and when a fact changes, the old row is marked Superseded and kept rather than quietly overwritten. The agents also read the cheapest thing first, so my profile page comes before a Key lookup, which comes before a filtered query, and a full search is the last resort.

Databricks put the principle better than I could in their [MemCurrent](#) post, where they write that "more history is not necessarily better memory", and on their own internal benchmark, adding memory tripled their agent's pass rate over having none, while using 31% fewer tokens and running 26% faster. My own numbers are a lot less flattering, since my Knowledge log has exactly five rows and not one of them has been superseded yet, which means the rule I'm proudest of hasn't really been tested. I'd still take five clean facts over 9,600 rows nobody can query.

My Feedback Is the Training Data

The agents don't get better on their own, and they definitely don't get better just because they have more memory. They get better when my complaints turn into rules that every one of them reads before acting.



The loop every agent runs on, whatever its lane.

When I told the agents to text like humans, that became `pref:bot-tone`, a `Stated` row in the Knowledge log that every agent reads. The first post Maya ever sent me was a list of my reasons for switching tools, and when I told her it had zero value for anyone reading it, that turned into a draft bar of one clear thesis, something a reader can actually use, sourced numbers and a supporting read from Janet, with Sunny checking every draft before it reaches me. Casing flipped three times, from lowercase to sentence case to Title Case on headings only, and the final version now sits in the voice guide with a small script that checks every draft against it, while the two dead versions of that script are still sitting in the folder like rings on a tree. I also had all 12 early blog drafts deleted, because every one of them was a tidy summary, and the bar since then has been one sharp original argument with firsthand data.

What all of that taught me is that memory gives an agent context, and taste is a separate thing entirely. Taste is still my job, and the loop is just how it slowly rubs off on them.

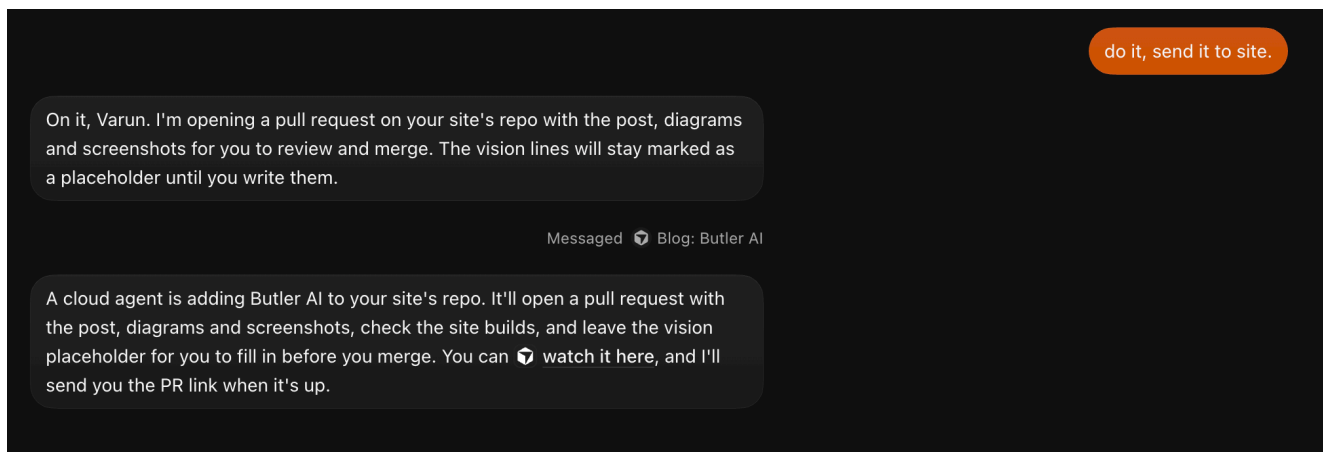
Why I Rebuilt My Stack

My tech stack keeps evolving, literally every week. Just about a week ago I ripped out my entire Claude subscription and replaced it with [the X Subscription](#), whose value proposition is something I greatly admire. For someone like me, based out of India, who spends money every month experimenting with paid APIs and tools, there's a real cost I need to factor in to justify this hobby, and this week I put every agent on lean credits, because running a butler isn't free and the butler has to know that as well as I do.

The other reason I rebuilt from scratch is that someday soon, I want my agents running my life entirely. I've always believed in [Human 3.0](#), which in my opinion is the way forward, because Maslow's hierarchy sets us all up for hedonistic failure and pushes us into the same herd-mentality decisions we then complain about. The X Subscription brings in context from code, conversations and concierge agents, all into one easy, manageable plane for me. X is truly the townhall of the world.

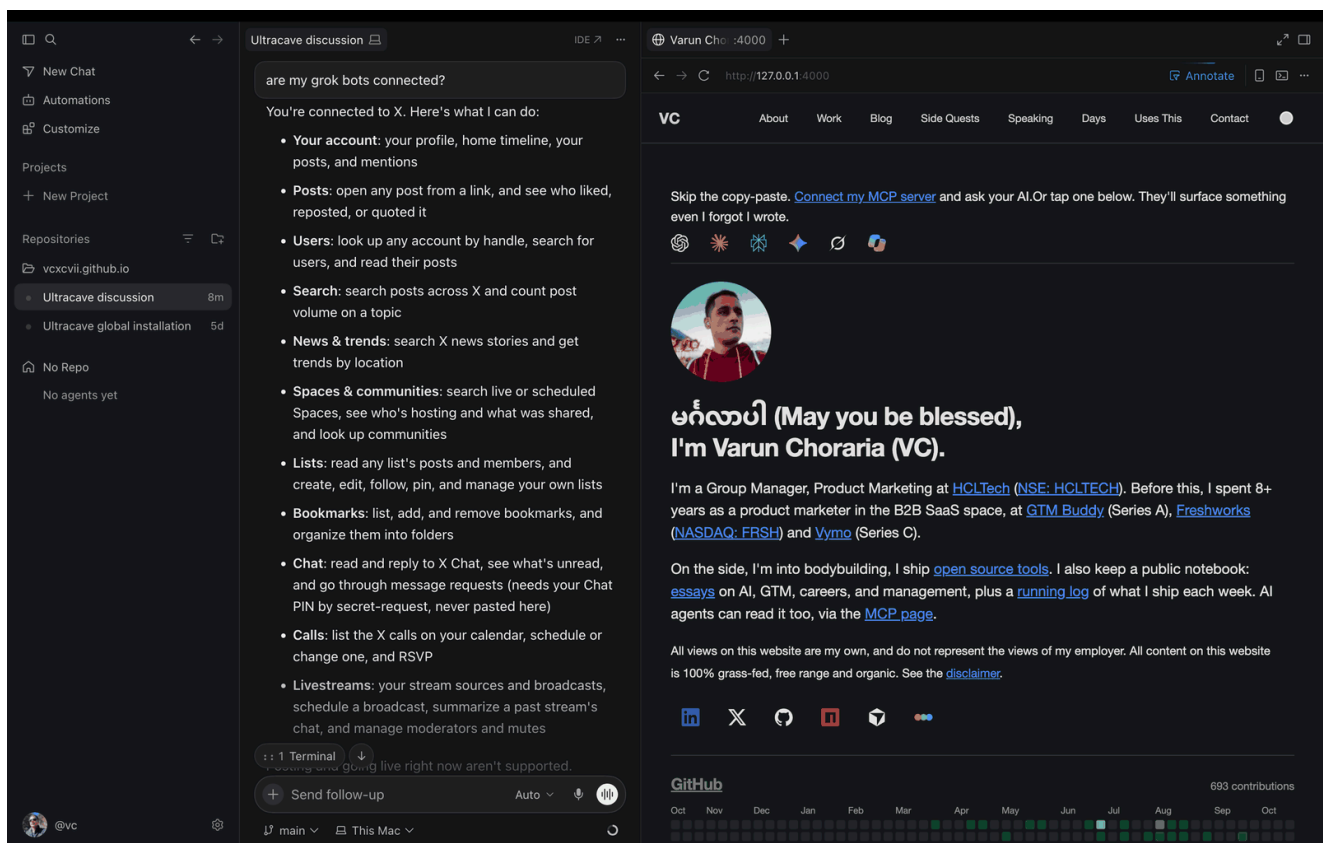
The Chat Is the Front Desk, the Repo Is the Back Office

That line about code, conversations and concierge agents sounds like marketing until you watch it happen on something you actually care about, and for me that was this post. Maya and I wrote it in the same chat where I ask her for everything else, and when I was happy with it I typed "do it, send it to site", which is roughly what I'd say to a concierge at a good hotel, and she handed the post, the diagrams and a short brief to a [Cursor](#) cloud agent working on [my site's repo](#). A card linking to that agent showed up right there in the chat, the agent opened a pull request, and the only thing left for me was to read it and merge it.



Maya handing the finished post to a Cursor cloud agent, which opens a pull request on my site's repo for me to review.

If you strip it down, every handoff between two people or two tools leaks something, and what leaks is almost always context, which is why the usual version of this job is someone copying a draft out of one app, pasting it into an editor, renaming images, fixing the front matter and losing a decision or two along the way. Here the talking layer and the doing layer sit next to each other, so the agent starts from the actual file and the actual brief rather than my memory of them, and nothing gets retyped. I still build most of this site in Cursor myself, with the same Grok Bots plugged in right next to it, but for this post I never opened an editor at all, and I suspect that's where this is heading, with the conversation becoming the interface and the IDE turning into a back office where the work happens out of sight.



My Grok Bots connected to X inside Cursor, with this site running right next to them.

It isn't free of cost or risk, though, and I'd be lying if I pretended otherwise. The cloud agent only knows what Maya hands it, so a lazy brief turns into a confident and wrong pull request, and there are now two products in the chain, each with its own limits and its own ways of breaking quietly. The bigger risk is honestly me, because a merge button is very easy to press without reading the diff, especially when the agent has already told you the site builds.

That's why the pull request is the one part I'd keep even if everything else around it changed. Nothing goes live on this site without my merge, and that single click is the most honest answer I have to the question of who owns the judgement.

Who Owns the Judgement

Janet keeps feeding Maya the same story from different companies, and if you line those reads up, the whole industry is quietly agreeing on what a personal agent actually is. Anthropic's pitch for [Claude Duet](#) is that "what makes Claude Duet so helpful is its judgment", and it checks your accounts only every few hours "because checking nonstop would use up your plan's limits much faster". OpenAI just made its [Decisions API](#) generally available, which does for their stack what Clef does for mine, answering bounded questions for routing and gates, and by OpenAI's own numbers it's up to 10x faster than sending the same question to GPT-6 Luna the usual way, at \$0.10 per million input tokens with no charge for output at all. Databricks' MemCurrent team says more history isn't better memory, and in [The AI Bill is coming](#), Patrick Wendell points out that the same model in a different harness can differ in cost by more than 2x, and that "token price alone is a poor predictor of what a model actually costs". So the real product, it turns out, is three boring things stacked together, cheap small decisions, curated memory, and restraint about when to speak up.

I agree with that, mostly, but I think it raises a few uncomfortable questions that the launch posts don't. The first is that the cheap decision call is going to be a commodity, and fast, because OpenAI's list price per token is already lower than [Clef's \\$0.24](#), and nobody should build a moat on a model that answers multiple choice. Wendell thinks there's real IP in routing, and for a company running thousands of engineers through one gateway he's probably right, but for a personal setup like mine, the value isn't in the model that picks the option, it's in who wrote the options, the 0.8 bar, the lanes and the list of things nobody's allowed to do.

The second question is who owns that judgement. Anthropic says it will keep tuning Duet's sense of when to message you "based on what testers tell us", which is fair, except the testers aren't you, and a vendor tuning restraint for the average user will always be a little too loud for some people and a little too quiet for others. My version breaks far more often, and the Fleet Bug Log is proof of that, but every rule in it is something I said, written down somewhere I can read and change.

The third is lock-in, and here Cloudflare is refreshingly honest, writing in their Agent Memory launch that "the more an agent learns, the higher the switching cost if that memory can't move with it." I could rip out Claude in a week precisely because my agents' memory lived in Notion, which no model owns, and if your butler's entire understanding of you sits inside one vendor's black box, you've basically signed up for a very polite form of hostage-taking.

So if you're building your own setup, here's where I've landed. Keep your memory in something you own and can export, keep the decision model behind one swappable call so you can switch the day something cheaper shows up, and write your interrupt rules down in plain words, because the model is the cheapest part of a butler now and gets cheaper every week, while the rulebook is the only part that's actually yours.

What This Is For

Which brings me back to Alfred, who never needed a bigger brain. He knew what to forget, and when not to knock, and that's the butler I'm trying to build.