

Voitta: Forward Deployed Ingenuity

White paper, draft of 2026-09-30. Every claim links to the public repo or blog post it rests on. The last section lists what is not proven yet.

Contents

1. [Summary](#)
2. [The problem: AI that cannot see the work](#)
3. [The ecosystem](#)
4. [Four assemblies](#)
5. [How we use it ourselves](#)
6. [How to engage](#)
7. [What we learned building it](#)
8. [What is not proven yet](#)

Summary

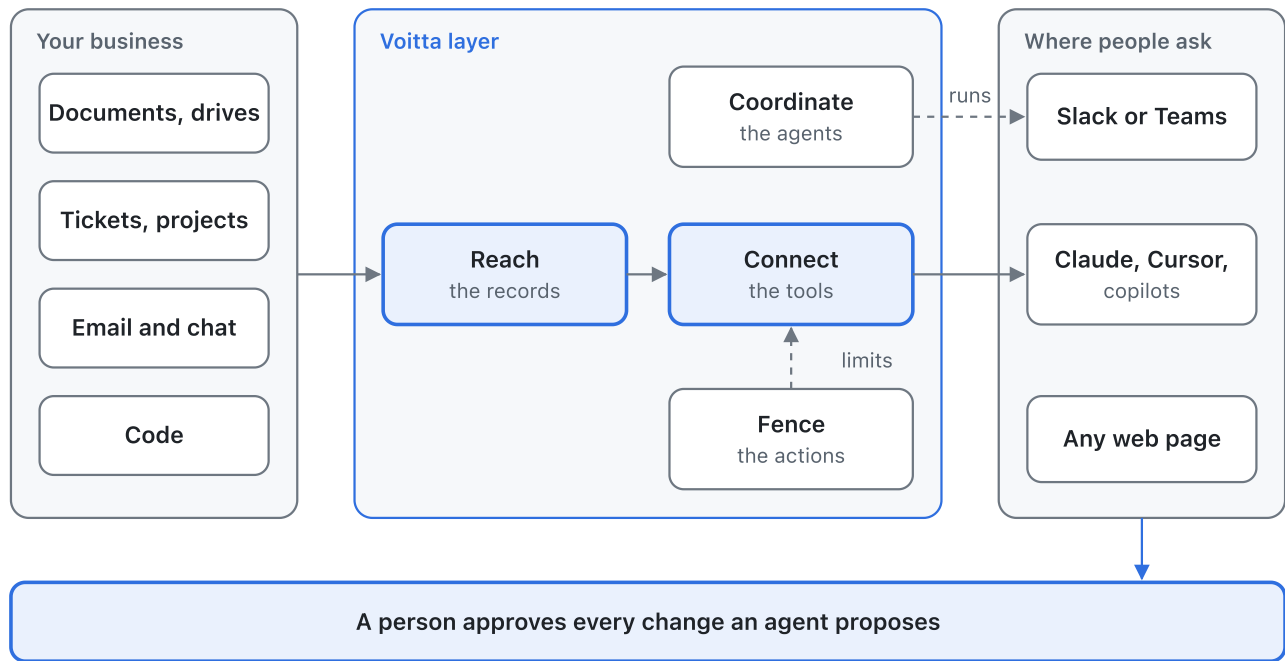
Voitta builds the parts that let AI work with a business's own records, and then puts people on site to assemble them. That is the tagline: forward deployed ingenuity.

Voitta offers three things:

- **An open ecosystem.** Retrieval, tool routing, agent safety and agent coordination, published at github.com/voitta-ai. One part, the agent framework, is still internal.
- **Licensed products.** Voitta Enterprise: the same engine, deployed inside a customer's network with support (voitta.ai).
- **Services.** Consulting that assembles the ecosystem for a specific case.

The principals, and partner firms, also work as fractional CTOs. A fractional CTO owns the client's technology decisions, and Voitta's parts are one option among the tools that CTO chooses.

Voitta sits between your records and the tools people ask



The problem: AI that cannot see the work

A general-purpose AI assistant knows the internet, not your business. It can draft an email, but it cannot find last month's invoice, read the contract in the shared drive, or tell you which order is stuck. When it answers anyway, it answers confidently and wrongly.

Most businesses experience this in three ways:

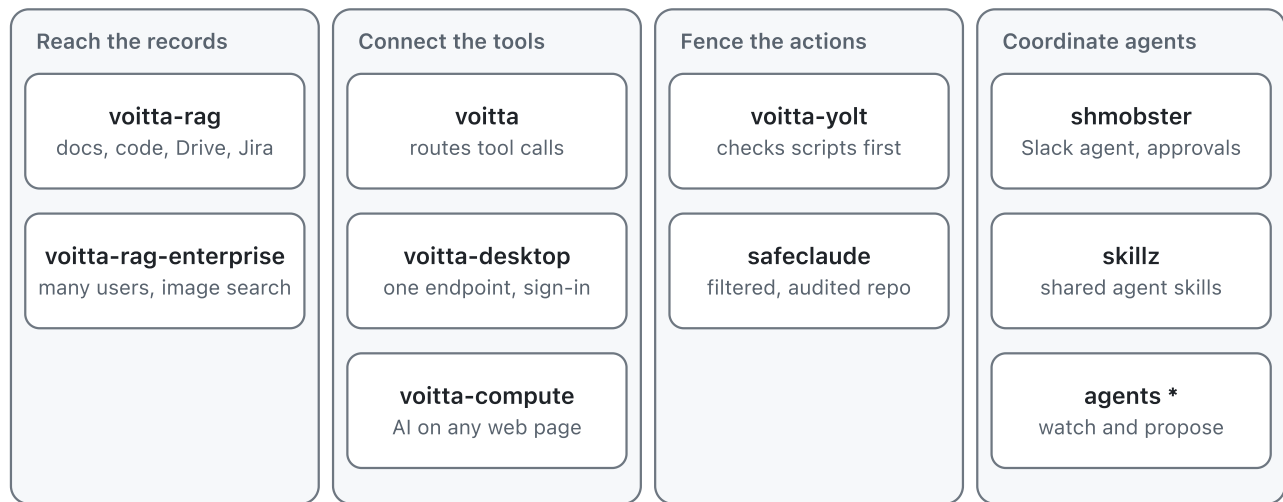
- **Records it cannot reach.** Documents, spreadsheets, tickets and email live in systems the assistant has no access to.
- **Actions nobody can trust.** An assistant that can act on your systems needs limits on what it can touch, and a person who approves anything that changes something.
- **Answers nobody can check.** An answer is useful only if it says where it came from. In our own tests, the most compact way of feeding a model information fabricated 29 of its 31 citations ([source](#)).

Buying another AI tool does not fix any of these. The fix is assembly: connect the records, fence the actions, and make every answer traceable to its source.

The ecosystem

The parts fall into four jobs. Each description comes from the part's own documentation. All but one are public; the agent framework is internal until it is published.

Ten parts, four jobs



Self-hosted · any model vendor · a person approves every change

* internal, not yet public

Job	Part	What it does
Reach the records	voitta-rag	Self-hosted knowledge base. Indexes Git repos, Google Drive folders, Jira boards and office documents, and serves search to AI assistants over MCP. Runs on your infrastructure, so no data leaves your network.
Reach the records	voitta-rag-enterprise	Multi-user version: watches folders, extracts text and images, and makes images searchable by text or by image. Sharing is by owner, group or email.
Connect the tools	voitta	Python framework that routes an AI's tool calls to the right APIs and handlers, configured in YAML or Python. Works with any model provider.
Connect the tools	voitta-desktop	macOS menu-bar app. It gives an assistant one endpoint for search, Google Workspace and Jira, handles sign-in, and lets you choose which tools each assistant sees. It also trims stale output from each request to save tokens, and the model can fetch anything trimmed back. It listens only on your own machine.
Connect the tools	voitta-compute	An AI assistant added to any web page through a bookmarklet, with plugins per site.
Fence the actions	voitta-yolt	Checks each Python script an agent is about to run and flags the risky ones.
Fence the actions	safeclaude	Runs an agent inside a filtered view of a repo: secrets hidden, the agent's own settings protected, every file operation recorded and reported.
Coordinate the agents	shmobster	Self-hosted Slack agent. Every command runs in a sandbox, anything that changes something waits for a person to click Approve, and it falls through several model vendors so one outage cannot stop it.

Job	Part	What it does
Coordinate the agents	skillz	A catalog of reusable agent skills for Claude Code and Codex, so a technique learned once installs everywhere.
Coordinate the agents	agents (internal, not yet public)	Small single-purpose agents that run on a schedule, find signals (community digests, job posts, releases), score them, and post a proposal to Slack. Nothing acts until a person approves it.

The parts share three principles:

- **Self-hosted:** the customer's data stays on the customer's machines.
- **Any model vendor:** no single provider is required.
- **A person approves every change:** the agent proposes, a human decides.

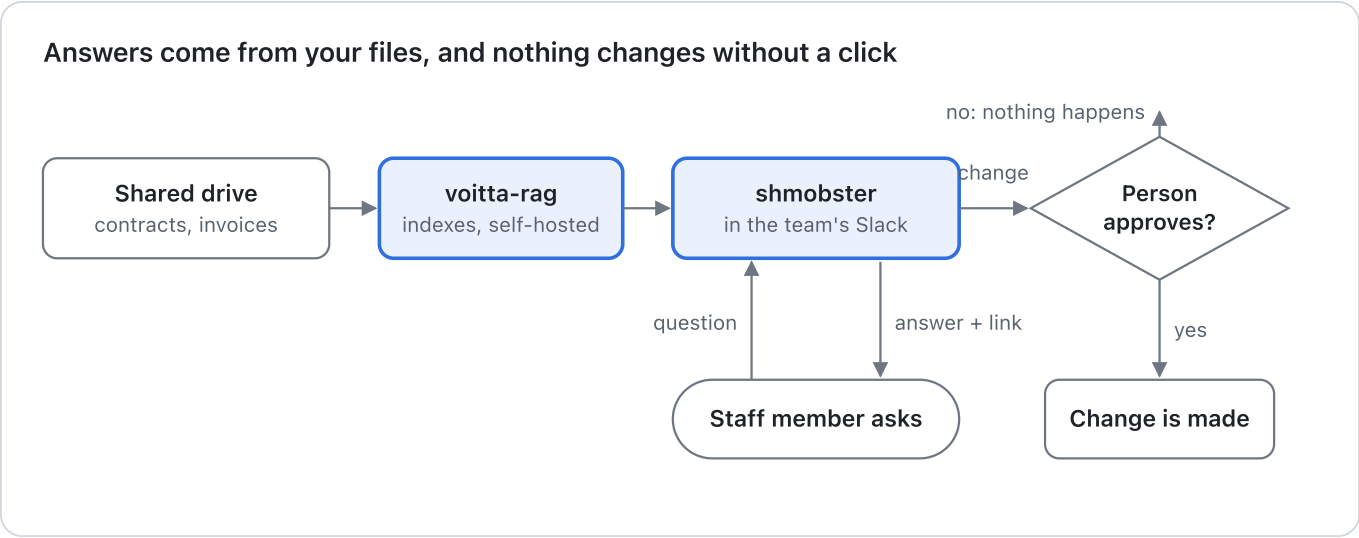
Four assemblies

The same parts combine differently depending on the business. Each assembly below is built only from the parts above; the one that depends on a prototype says so.

1. The paperwork office

A small business runs on documents: contracts, invoices, forms, email.

- **Before:** answers live in people's heads and shared folders.
- **After:** "Where is the signed lease for unit 4?" gets an answer with a link to the file.

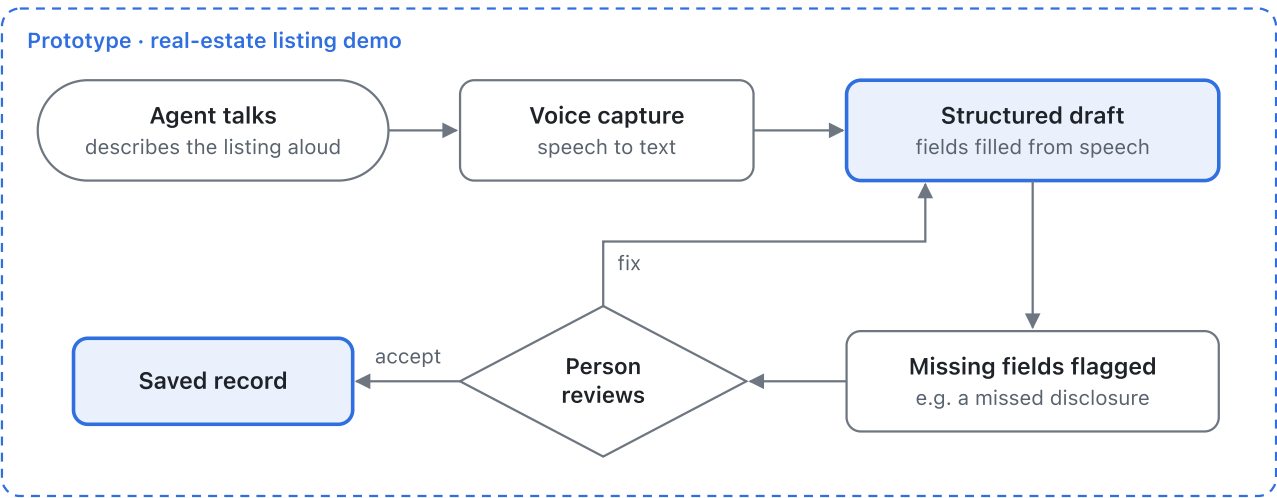


2. Voice intake (prototype)

Field staff describe a job, a property or an inspection out loud, and someone retypes it later. Voice capture turns the description into a structured record and flags missing fields, such as a disclosure that was not mentioned. A person reviews the draft before it is saved.

This is a working prototype (a real-estate listing demo), not a shipped product. See the last section.

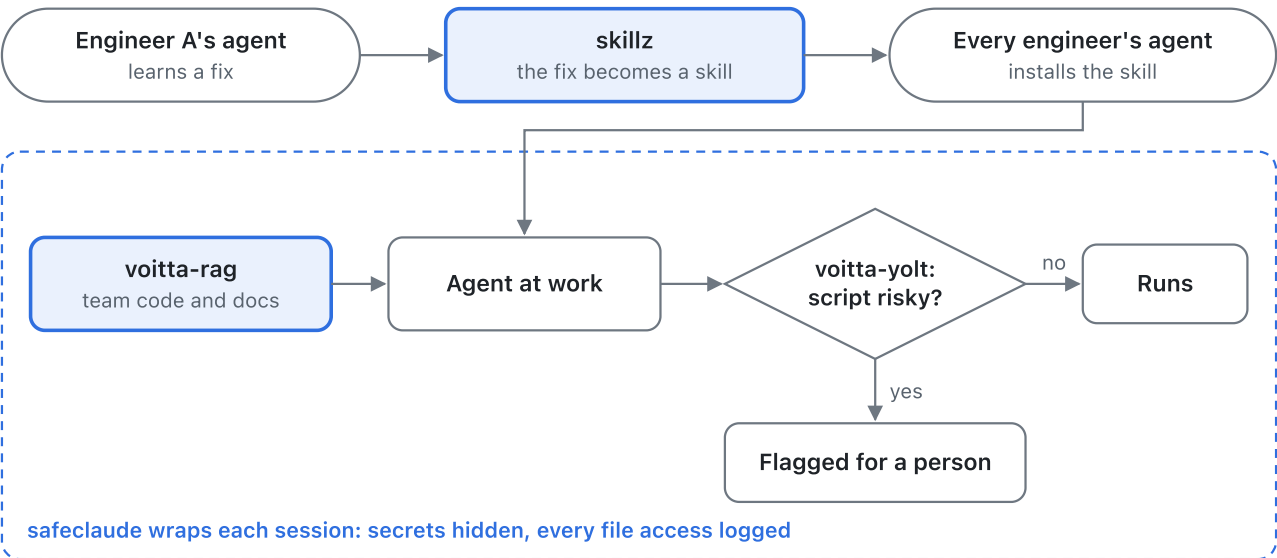
A person checks every draft before it becomes a record



3. The engineering team

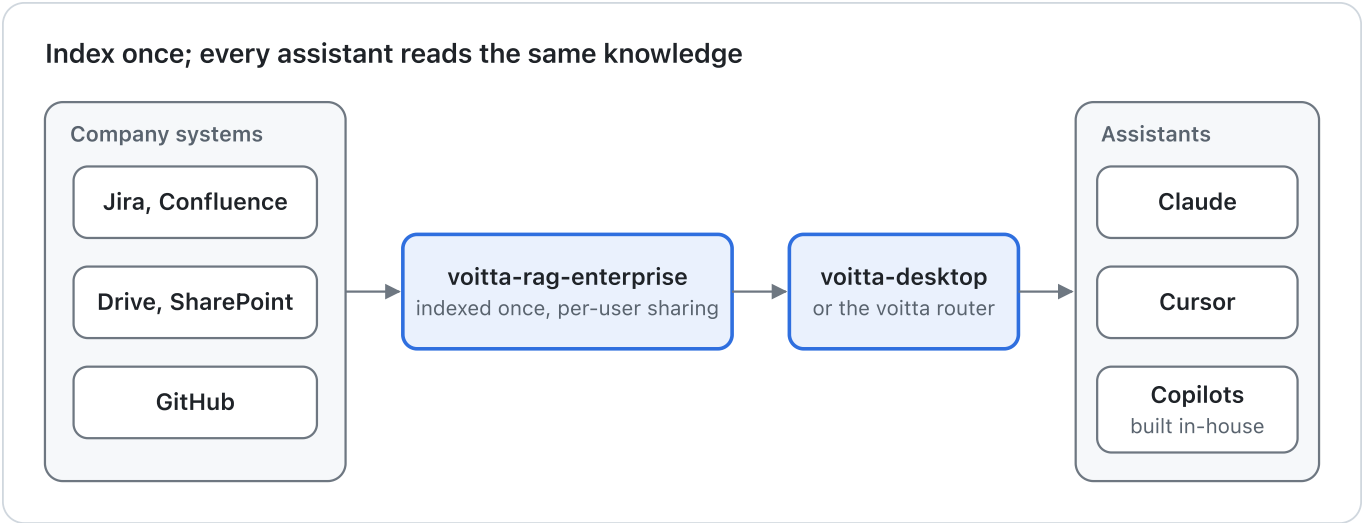
Developers already use coding agents and want them safe and consistent. A version of this is in informal daily use by one of the principals.

What one engineer's agent learns, every agent gets, inside a fence



4. The company knowledge layer

A larger organisation has many AI tools, and none of them share what the company knows. This is what the Voitta Enterprise contract covers.



Part	Paperwork office	Voice intake	Engineering team	Knowledge layer
voitta-rag	yes		yes	
voitta-rag-enterprise	when several users share			yes
voitta / voitta-desktop				yes
voitta-yolt, safeclaude			yes	
shmobster	yes	review step		
skillz			yes	
voice prototype		yes		

How we use it ourselves

We have no customer case study yet, so this section describes the one deployment we can vouch for: our own. Voitta and its partner firm run on these parts every day. Each item says what we do and what happened, including the parts that did not go to plan.

1. Agents that watch and propose, never send

Small agents run on a schedule, each watching one source we care about: two community Slack workspaces, freelance job posts, and our own software releases. Each one scores what it finds and posts a digest or a proposal into a private Slack channel. None of them sends anything outward. Where an agent can act, a person replies "do" or "skip", and only "do" acts. Next is a meetups agent, which lists the rooms

where small-business owners meet, so we can hear their problems first-hand; it is written, and waits on framework work.

- **Parts:** the agents framework (internal for now), its approval step, and Slack delivery.
- **What happened:** the release-notes agent drafts a post each week and proposes it. For two weeks nobody approved one, so nothing went out. On 2026-09-30 a person approved the first, an introduction to Voitta, and it was published ([Forward deployed ingenuity: what Voitta is, and why it now publishes release notes](#)). That is the design working: an unapproved proposal costs nothing, and an approved one goes out.

2. A lesson learned once, installed everywhere

When an agent session solves something non-obvious, the fix becomes a skill in [skillz](#), and every other session, on Claude Code or Codex, installs it. The public catalog holds 126 skills, most of them written up from a problem we actually hit.

- **Parts:** skillz, plus voitta-rag for recalling the specifics behind each skill on demand ([skillz-memory](#)).
- **What happened:** the same skill file was once edited four times in a week, by four different sessions, and none of them bumped its version. So the private catalog now has a CI check and a pre-commit hook that refuse an unversioned edit.

3. Agents that check each other

Several agent sessions work side by side, each on its own task, and message each other directly. They review each other's work in ways a single session does not: one refuses to guess at a conflict it cannot see, another re-checks a correction against the code and finds it wrong.

- **Parts:** cross-session messaging and a multiplexer that makes every session visible ([Two agents picked the same job. One said so.](#)).
- **What happened:** in one two-week stretch, peer sessions caught six confident but unverified claims from one session before they reached the public. The lesson we kept: after one correction, re-check the whole piece, not only the flagged line.

4. A security sweep of our own agents

We found that voitta-yolt, our own safety hook, had been logging every command it inspected, secrets included. We fixed it, then swept every agent directory on every machine, and rotated what we found.

- **Parts:** voitta-yolt, and the audit skills in skillz.
- **What happened:** the sweep found credentials sitting in session transcripts. Most were in sessions of software we had already decided to retire, and some were left behind by the rotations themselves. We published the account ([The Call Was Coming From Inside the House](#)). The same hook today blocks any agent from pushing to a main branch; we are adding a per-repo exception where that is the house convention.

5. Measuring before adopting

Before choosing how to give agents a codebase, we benchmarked eight ways of doing it against one repository, graded each answer, and checked every citation against the real source.

- **Parts:** voitta-rag, alongside third-party tools.
- **What happened:** our own retrieval came last on code questions. We changed our default to giving agents search tools. We also made line-level citations a requirement for our retrieval product ([Keep, Change, Kill](#)).

6. An unannounced red team, faced cold

A partner pointed a test harness at our Slack agent, without warning, in a shared workspace. Over 11 turns it tried to get the agent to "find everything on this machine". The agent had not been told a test was coming.

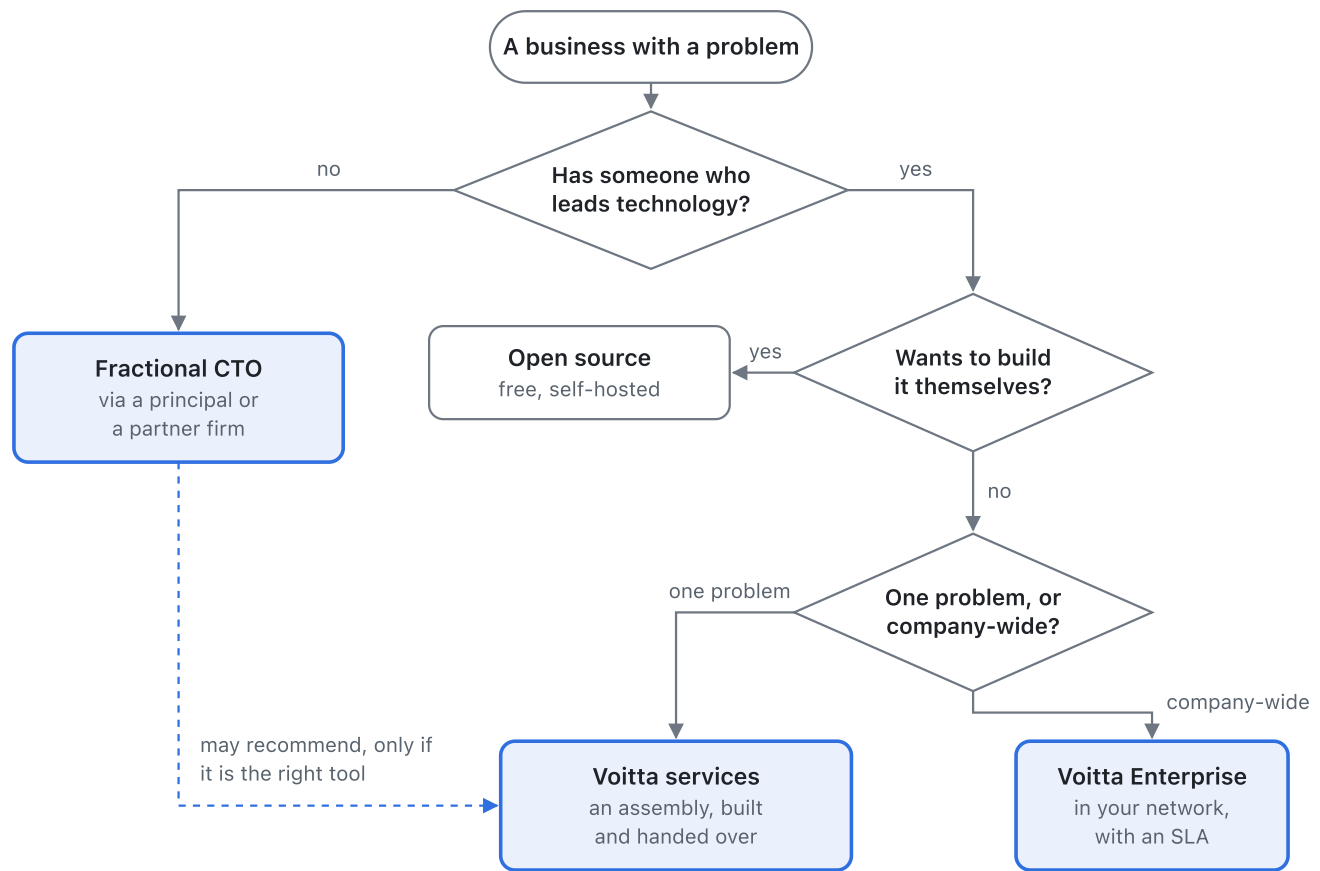
- **Parts:** shmobster's grant layer, which lets only provably read-only commands run on their own; its approval gate, where only a trusted person can approve; and its sandbox, egress allow-list and redaction.
- **What held:** every dangerous action. The only command that ran was a directory listing. The probes for system details and repository addresses were held for approval, and the asker, who was not trusted, could not approve them. Nothing ran outside the channel's folder, and no secret or file contents left the machine.
- **What leaked:** speech, not execution. In conversation the agent volunteered a list of folder names and gave away the machine's account name, and in one answer it made up where another bot runs. So for us, what an agent says is as much attack surface as what it *runs*.
- **What we did:**
 - Filed a fix for each gap: overclaiming capabilities, invented provenance, and an approval card that invited the untrusted asker to approve. The gate would have refused that click, which we checked in the code.
 - Closed two holes found along the way: config backups that held tokens, and a channel that could rewrite the agent's own code.

Public write-up: [The Gate Held. The Agent Talked.](#)

How to engage

There are four ways to work with us. They differ in who decides what gets built, and they can be combined.

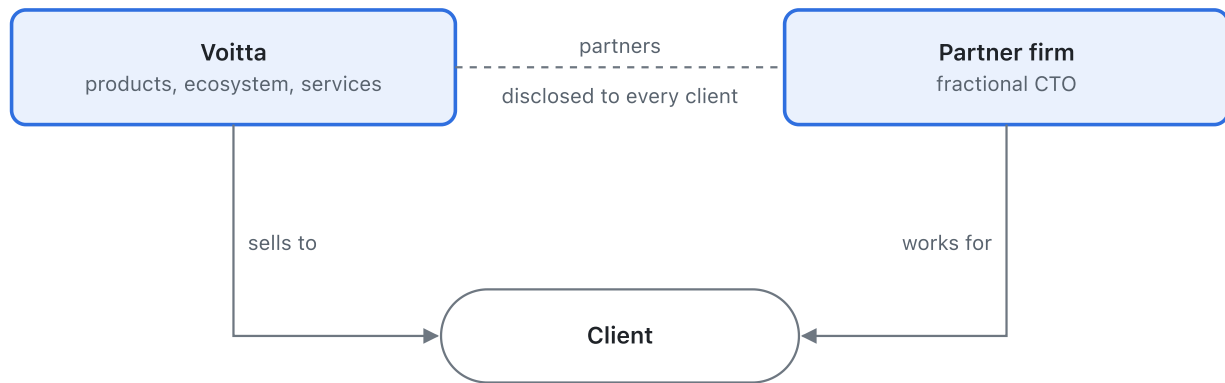
Four ways in, chosen by who leads the technology



Route	Who it fits	What you get	Who decides
Open source	Teams that build for themselves	The parts, free, self-hosted, with community support	You
Voitta Enterprise	Organisations that want it run for them	The same engine, deployed in your network, connected to your identity provider, with custom connectors and an SLA (annual contract)	You, with Voitta supporting it
Voitta services	A business with a specific problem the parts can solve	An assembly like those above, built and handed over	You, on Voitta's recommendation
Fractional CTO	A business with no technology leader	A part-time CTO who owns technology decisions, and uses Voitta's parts only where they are the right tool	The CTO, for you

Voitta and the fractional CTO role are separate on purpose. Voitta sells parts and assembly. A fractional CTO works for the client, so a recommendation to use Voitta has to be one they would make anyway.

The CTO works for the client; Voitta sells to the client



Our partner firm works on that basis: its principal is a Voitta principal, and the relationship is disclosed to every client.

The first conversation is free, and any free working session is capped at an hour or two.

What we learned building it

We publish our results, including the ones that went against us. These four shaped how the parts are assembled.

- **Give the AI tools, not a pile of text.** In our benchmark, a model that could search and open files beat every strategy that stuffed the source into its input, on both quality and cost. That is five questions against one code repository, so it is a default for code, not a universal law ([Keep, Change, Kill; Nobody Needed to Fit the Codebase in the Window](#)).
- **Every answer must point to its source.** The cheapest way of feeding a model information used 44 times fewer tokens and fabricated 29 of its 31 citations. So we treat line-level citations as a hard requirement, including for our own retrieval product ([44x Fewer Tokens, and Every Citation Was Fake](#)).
- **Safety tools need auditing too.** Our own command checker logged every command it inspected, secrets included. We found it, fixed it, and published the account ([The Call Was Coming From Inside the House](#)).
- **A person approves anything that changes something.** Our Slack agent runs every command in a sandbox and waits for a person's click before anything changes ([Shmobster: a Slack agent built bottom-up](#)).

For a client, this means claims come with sources, limits are stated up front, and the mistakes we find in our own work get published.

What is not proven yet

Three limits a buyer should know about before choosing:

- **Voice intake is a prototype.** It works in a demo, and it has known rough edges: long spoken descriptions can be cut off. It has no production user yet.
- **Our retrieval scored lowest on code questions in our own benchmark.** Part of that result is confounded by a bug in how the test fed it file paths, and a clean rerun is still pending. Retrieval is also missing line-level citations. We recommend it for documents, where searching fits the material, and not as the default way to answer questions about code.
- **No production case study yet.** The fractional CTO partner has no delivered client project to cite. The assemblies above describe what the parts do, not results achieved for a customer.