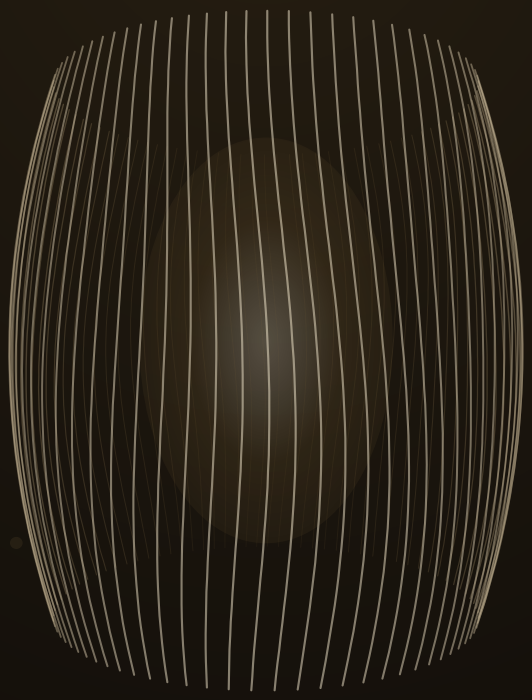
UNDERSTANDING WEAVE



Weave: the *architecture*.

A technical account for the reader who will not take it on faith. What is built, what is staged, what is designed, and how it is made to get better on its own, once it runs, without ever leaving the boundaries it is allowed.

WEAVEDEEP.COM

THE TEST

Start by assuming the worst about it.

The honest way to read a document like this is to assume the thing it describes is less than it claims, and to look for the seam where the claim and the reality come apart. So begin with the sharpest version of the suspicion: that Weave is a language model with a search index bolted to the front, a retrieval-augmented chatbot wearing the costume of a system. It is the right suspicion, because most of what is sold as enterprise AI is exactly that, and the only answer worth giving is one you can check.

A language model is a component Weave uses, the way a car uses an engine. It is not the car, and the difference is the entire subject of this paper: how knowledge is held, how an answer is grounded, what the model is and is not permitted to see once sealing is enforced, how its output is checked before it reaches you, and, the part that matters most over time, how the whole arrangement is designed to get better at each thing the moment it learns it, and to keep improving on its own, within boundaries it is never allowed to cross. That last part is a design, and this paper is careful throughout to hold it apart from what runs today.

Throughout, one discipline is kept without exception. Everything running today is written in the present tense. Everything built but deliberately not yet switched on is marked BUILT · STAGED. Everything decided and designed but not yet built is marked DESIGNED. And the design constraints that shape every choice, rather than any running service, are marked PRINCIPLE. A system that cannot tell you, to the file, which of these each of its capabilities is, is a system you should not trust. Weave can, and the labels below are that accounting.

THE SHAPE

One read-only intake, a structured memory, a mind that checks its answers.

Weave runs as a single product, a Node and Express service, with every operating number held in one table and printed at boot so the running configuration is never a guess. A client shares a Google Drive folder with Weave's service account, read-only. Intake walks it, holds a first conversation about what it found, and, once the scope is agreed by a person, reads the documents and writes what they establish into Memory: a set of Notion databases that belong to that brain. The Mind reasons over Memory when a person asks, and can make artifacts from it. A living surface is what the person actually meets. Around these sit the organs that keep the system honest about itself.

The path of a fact: from a read-only Drive, through intake, into a per-brain Memory, reasoned over by the Mind, met at the surface. The lower row is what watches the running system.

Two things in that picture are load-bearing and easy to miss. The arrow from Drive to intake points one way only: Weave reads through a service account scoped to view, every call is a GET, and nothing is ever written back to a client's files. LIVE TODAY And the step where a person agrees the scope is not a formality; the separation of one company's material from another's is decided there, by a human looking at the evidence, never by a similarity threshold that could quietly merge two clients.

LIVE TODAY

MEMORY

A structured record, not a pile of text.

Weave does retrieve, and the difference from ordinary retrieval is the whole point. An ordinary retrieval system keeps your documents as searchable text and hands a model the fragments that look relevant. Weave instead reads the documents at intake and writes what they establish into Memory as structured records, held in a set of Notion databases that belong to the brain, clients, projects, decisions, commitments, signals, and context, with two-way relations between them and a Soul page for who the organisation is. LIVE TODAY

The unit of truth is therefore a record with relations, a decision linked to the people who made it, the project it governs, and the date it was taken, rather than a chunk of prose whose relevance is a guess. Memory is addressable, queryable, navigable by a human being, and correctable: every write Weave makes lands here, where it can be read, fixed, and carried away. The claim is not that Weave is not retrieval: it retrieves over a structured, relation-linked record — extracting entities into a graph is a known family of technique, not an invention — and then, as the next section shows, checks every answer against that record before letting it go. The two together, a record a person can correct and a verifier whose strength is contradiction with that record, are what set it apart from a chatbot with a good index; neither alone would. Neither a structured record nor claim-level checking is new in the research literature; what Weave claims is not novelty of the parts but that both run over a store the organisation owns and can correct.

Two details show the grain of it. Dates are held in three states and never flattened: certain, ambiguous (where every reading the source allows is kept, rather than one silently chosen), or absent. LIVE TODAY And the words Weave uses are the organisation's own, learned from its own record over a fixed set of underlying kinds, a party, a person, an obligation, a decision, an event, so that a hospital and an agency each meet a Weave that sounds like it grew up inside them. That vocabulary layer is built and runs in shadow, informing nothing a person sees until it is ruled ready. BUILT · STAGED

Because a brain can hold its own workspace credential, Memory can live inside the organisation's own Notion, under the organisation's own account. It is theirs in the plain sense: Weave reads the organisation's files and writes only here, to a store the organisation owns and could keep, intact, the day Weave is gone. LIVE TODAY

THE MIND

It answers, then checks the answer against the record.

When a person asks something, the Mind does not simply prompt the model and relay what returns. It retrieves the records that bear on the question, scored for relevance against a fixed token budget, with coverage, genuine disagreement, and any records the question names pinned so they cannot be crowded out, interleaved across the databases so none dominates. It builds the system prompt from

that. The model answers. And then the answer is checked, claim by claim, against Memory.

LIVE TODAY

The answer is not the last step. Every claim is verified against the record; a failing answer is revised once, and whatever still fails is stripped rather than shipped.

The checkers are rule-based and specific: anchors (dates, money, months, percentages), ownership, disagreement, undisclosed conflict, denial, ordering, and derived attribution. They are high-precision rules, not oracles. A paid semantic pass runs on top when an answer carries the kind of thing a fixed rule cannot safely judge, figures, dates, commitments or cross-document links, and above all the negative and the completeness claims. Whatever fails is sent back for a single revision and checked again; whatever still fails has its sentences cut, and if nothing coherent is left the answer says which records it holds and asks you to open them rather than take its word. LIVE TODAY

What that buys, and what it does not. It catches the faults it can define against the record: a wrong date, a misattributed figure, an order the record contradicts, a completeness claim the record will not support. It is a strong guard against contradiction with what is stored. Its boundary: a fluent claim that contradicts nothing stored and trips no anchor can still pass, and the hardest cases, the negative and the completeness claims, are a model judging a model, with the correlated blind spots that implies. And the loop checks the answer against Memory, which was itself read out of the documents by a model at intake, so what it guarantees is consistency with the extracted record, not with the original page. A separate instrument exists for that gap, asking whether the memory is true against its sources.

LIVE TODAY

Within those bounds, this is what keeps a claim from being dressed as more certain than it is. Everything Weave tells you sits in one of three states and is labelled: proved, where the record determines it and Weave can name the basis; unresolved, where the source raises a thing but does not settle it, and Weave carries the source's own words; or absent, where the record is silent, and silence is not read as the absence of the thing.

One more edge. The checker verifies the answer; it does not yet interrogate the question. A false premise carried inside a question, if it trips no ordering or anchor check, can still frame an answer that is locally true. A check that tests a question's own assumptions against the records is built and shipped switched off, behind a single setting, awaiting the decision to turn it on, which is itself a human gate.

BUILT · STAGED

The Mind has a second form. Besides answering, it makes: a synthesised artifact built from Memory and left behind, owned by the organisation rather than reasoned once at question time and lost. Sealed records are excluded before that artifact's prompt is built and checked again on the way out, and everything it makes needs a person's decision before it moves anywhere in the world. LIVE TODAY

PROVENANCE

Every output is meant to carry how it was made, stamped by the pipeline.

A system that grounds its answers has to answer a second question: grounded by what, and produced how. Weave's design requires that everything it produces carry its own provenance, the model, the provider, the model and policy and prompt versions, a timestamp, and an integrity hash. PRINCIPLE

```
model   resolved from the routing policy, never named in code
provider   whichever one served it
versions   model · routing-policy · prompt
time   when it was produced
hash   sha256( the output )   computed after generation
stamped by   the pipeline, not the model
```

Two design choices give this teeth. The stamp is written by the pipeline, not by the model, because a model's report of its own deployment identity is unverifiable and worthless as evidence, while the pipeline records it from outside. The design also calls for a hash taken over the output after generation, so that a later change to the stored text would no longer match its record. The strength of such a hash is narrow and worth stating: held in the same store, writable by the same credential, it would detect accidental corruption and out-of-band edits but would not be tamper-proof, because whatever can rewrite the output can recompute the hash. Real tamper-evidence takes an append-only log, and binding an output to a named provider takes signed attestations; both are named here as future work, not claimed as present. What such a hash would prove is integrity within Weave's own trust boundary, that this stored text still matches this stored record, not that a named provider truly served it. PRINCIPLE

Where this stands. One piece already runs: on the operator's side a counts-only row per answer records, by default, which model answered it, so the fact of which model produced which answer is live, held beside the brain rather than ridden on the output. LIVE TODAY Built but not yet switched on is the stamp proper, provider, the version triple of model and routing-policy and prompt, and a timestamp, written onto every output itself; it is carried on a branch and covered by the test suite, held there until after the first live pilot. BUILT · STAGED The integrity hash over the output is further out still: it is designed, not yet built in any branch, which is why the stamp above the record block is a principle rather than a running guarantee. An architecture that stamps provenance is worth only the day the stamp rides every output, and that day has not arrived.

SENSITIVITY AND REVERSIBILITY

Sealed before the model sees it *under enforcement*; by default, flagged and left in place.

The usual approach to confidential material is to let the model read everything and ask it, politely, not to repeat the delicate parts. Weave is built to do the opposite: the design seals sensitive substance out of the material before the model sees it, so that under enforcement there is nothing for a clever prompt to coax back. PRINCIPLE

This is the kind of place a claim slips, so here is what runs today. In its default posture the seal runs in report mode: it detects what it would withhold and names it, but leaves it in place rather than blocking it, until an operator turns enforcement on. So by default the sensitive substance is flagged, not yet removed; the remove-before-the-model guarantee holds under enforcement, which is a deployment decision rather than the out-of-the-box state. LIVE TODAY Where a deployment sets owner-open, the owner of a sealed matter can open their own. And Weave measures what its own sealing misses, including the subtle case where a sealed figure can be reconstructed from the open ones beside it, rather than assuming the seal is perfect. LIVE TODAY

Underneath sits reversibility, a precondition rather than a feature. Every change Weave makes is logged with its reasoning and can be undone within the stores Weave controls: a corrected fact does not sit beside the wrong version it replaces, and a whole run can be rolled back. The bounds are two: reversal is per run rather than per individual record, and once something has passed out of Weave's stores into a backup or an external system, Weave no longer governs its deletion. LIVE TODAY Within those two bounds, the cost of a wrong move stays bounded and visible.

SUBSTRATE INDEPENDENCE

The model is built to be a *replaceable part*; today one provider still carries the load.

A product built on one model is mortgaged to that model's release schedule. Weave's constitution forbids it. No model is ever named in a prompt or a scenario; a versioned routing policy maps each kind of task to an ordered set of models that have each passed Weave's own evaluation for that task, and only a model that has passed ever runs. When one retires or is unreachable, Weave falls to the next proven model; if none is available it stops and says so rather than running something unproven. Stability, here, does not mean the output never changes; it means the output always clears the evaluated quality floor and, with provenance recorded on every output, is reproducible from it. PRINCIPLE

What runs today is the smaller version of that, and none of the three commitments above, eval-gated routing, provenance on every output, and proven failover, is live in production yet. Production calls a single provider, with the model supplied by configuration and never written into the code, failing loudly if that configuration is absent, so the seam is real even though the machinery around it is not switched on. That machinery, one module that asks for a job and not a model, the routing policy shipped inside the product, failover that can never reach a model that failed its test, and a second provider already adapted and carrying no traffic, is built and held on a branch until after the first live pilot. BUILT · STAGED

Beyond that is the part that makes this compounding rather than merely safe: the crucible, in which each candidate model must earn each job by passing its evaluation, under a fixed monthly ceiling, so that moving to a newer or cheaper model becomes something Weave does on evidence rather than a migration you pay for. DESIGNED

HOW IT IS BUILT TO GET BETTER

Designed to improve itself, and standing still on purpose.

First, plainly: today Weave does not improve itself autonomously at all. It checks with a person on essentially everything. What this section describes is how it is designed to improve, and the fence that improvement will always run inside, not a loop that is spinning now. LIVE TODAY

The design, then. Weave's constitution does not treat improvement as a roadmap item; it makes continuous improvement the default, so that any process, prompt or routing rule is a standing candidate to be made better, and a change that is possible and breaks no constraint is meant to be effected rather than filed for someday. A system built to that principle is not static by nature; the intent is that its behaviour a year from now is not its behaviour today. PRINCIPLE

Three mechanisms are meant to carry it, and which exist today is worth stating. A lesson is to be learned once, as a mechanism rather than a note, so a fix made in one place is inherited everywhere it

applies and Weave grows immune to its own stale assumptions instead of rotting as the facts age; this is designed, not built. DESIGNED The model substrate is to manage itself, sensing the field, evaluating new candidates, and drafting evidence-backed changes to its own routing; also designed, not built. DESIGNED What is live today is narrow and real: a correction you make lands in Memory, and the next answer reasons over the corrected record. That is the corrected record being read the next time, not the compounding self-improvement the first two mechanisms describe. LIVE TODAY

The improvement loop, once it runs, is designed to sit inside a fence that does not move: a person approves what is adopted, every change is logged and reversible, cost is capped. Preparation is machine work; promotion is human work.

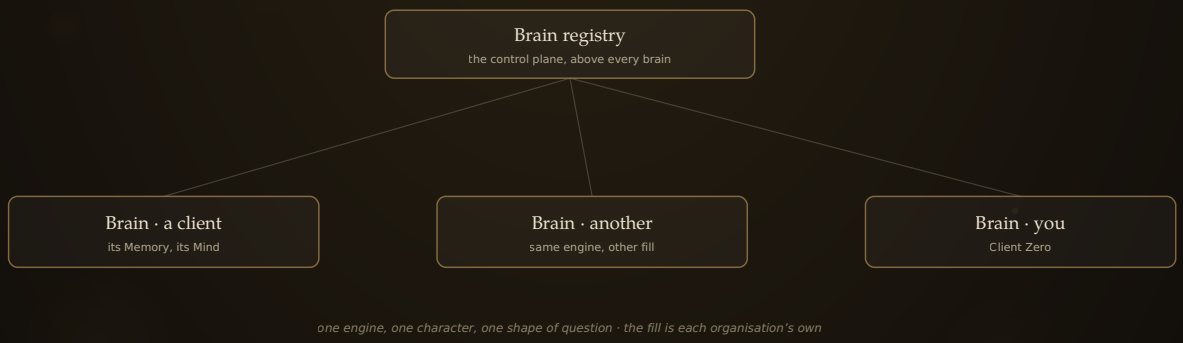
The fence, because a self-improving system without one is the thing to fear. Five changes are reserved permanently for a person, and no amount of the system's own cleverness may take them: the review of what is sensitive, the promotion of any new method or model into production, staging, the decision to accept or reject, and any write to a real brain's memory that has not been approved. Now the mechanism, since it is the point: today only one of these, the promotion of a made artifact, is enforced in code, and even that authenticates a brain's key rather than a named person; the other four are held by operator discipline and credential separation, not yet by a code gate. The constitution names them permanent; turning each into enforced code is work still in progress. Around them, every change must be reversible, logged with its reasoning, and visible, and autonomous spending is tiered and capped, stricter inside a client's environment than in Weave's own. PRINCIPLE

So the true state of this moment: the capacity to improve is constitutional and partly built; the permission to exercise it unattended has not been granted. Autonomy is a ladder Weave climbs only as it earns trust, at a pace its owner sets, and it stands on the bottom rung by design, where a person still signs off on essentially everything. That is the safety model working as intended. LIVE TODAY

ONE BRAIN, AND THEN MANY

Built once, stood up *per organisation*, under one control plane.

Weave was designed from the first line to be built once and stood up many times. The engine, the character, and the structure of a question are identical in every brain; what differs is the fill, your records, your language, your decisions, which is yours and unrepeatable. PRINCIPLE



The designed topology: a control-plane registry above every brain, each the same engine with a different organisation inside it. The registry format exists; many brains learning from one another is designed, not yet built.

Above any single brain sits that control plane: a versioned registry naming every brain Weave manages, its identity, the Memory and artifact databases that are its stores, the name of its credential rather than the credential itself, its voice and Drive folder, and its state. It is kept deliberately above the brains, so that archiving one never orphans the index. That registry is built and in use, with more than one brain registered and opened through it today. In the registry, adding a brain is appending one entry, but the real work of standing one up, the client's own workspace credential, the read-only Drive share, the scope agreed in the first conversation, and each action-model earning its place by evaluation, is deliberately not hidden behind that line. LIVE TODAY

What is not built is federation itself, many brains actually learning from one another. Each registry row today holds a brain's identity, its stores, its credential pointer and its state; nothing in it yet carries a lesson from one brain to the next. The reason it is unbuilt is its own kind of discipline: premature federation is complexity bought before it is needed, and Weave does not fake what it has not earned. DESIGNED

WHAT IS PROVEN, AND WHAT IS NOT

A system that can show you its own edges.

Everything above divides into what runs, what is built and staged, and what is designed, and those labels have been kept without exception. Weave holds a rule unusual in this field: no capability is called proven without a named artifact that proves it, and a value is not treated as real until it is a behaviour with a failure mode and an instrument that would catch it failing. PRINCIPLE

That rule cuts against Weave's own story. Weave publishes a computed account of itself that marks, for each of its self-claims, whether an instrument is watching it or whether nothing yet is, and names the instrument each uninstrumented claim still needs, rather than letting a claim pass on the strength of a sentence. How many are still uninstrumented is a number that lives in that computed account, not frozen into this page, precisely so this page cannot quietly go stale against it. LIVE TODAY The account also keeps two kinds of evidence apart: what can be measured on live data, that it put no wrong thing in, is checkable without an answer key and travels to every deployment; what it may have missed, that it left no right thing out, can be measured only against a known answer key in a lab. The first is never allowed to stand in for the second. PRINCIPLE

Its edges are named here so that reaching for more never becomes claiming it: inferring what a person cares about from behaviour is unproven, judgment beyond the records is where the value and the risk both live, and trustworthy is not permitted to drift into omniscient.

So, to the reader who began by assuming the worst: keep assuming it, and check. What has been described is not a finished machine but a real one, built on purpose, specific to the file about what it does and does not yet do, and designed so that what it learns it keeps and improves on, inside a fence it may not cross. That is not a language model in a costume. It is a correctable, contradiction-checked, organisation-owned record with a model reasoning over it under human control, early in its life, and the labels above are where to test every word of that.