SYSTEMS NOTE · AGENT ARCHITECTURE

Anatomy of a self-hosted agent OS

A production system that reads everything a business produces, keeps a memory that outlives every conversation, acts on its own schedule, and re-examines its own conclusions overnight — with customer data never leaving the building.

Prepared for discussion

Architecture and operating principles. Client identifiers and internal metrics generalised.

2026

Three temperatures

Every component in this system is one of three things: work that runs on owned hardware at no metered cost, work that calls a frontier model and is billed per token, or deterministic code with no model involved at all. The colour coding below runs through the whole document. It is the most important thing on the page — most of what looks like AI here is not billed, and the parts that are billed are deliberately small.

LOCAL On-premises models. No metered cost, no data egress.

CLOUD Frontier reasoning. Metered per token.

CODE Deterministic. No model involved.

The expensive model is a small part of the system, used late.

Transcription, embedding, fact extraction, consolidation, deduplication and first-pass drafting all run on local models on owned hardware. Frontier reasoning is reserved for work that genuinely needs judgement. That keeps the metered surface small, the capability large, and — for any business holding regulated customer information — keeps the sensitive corpus inside the perimeter by construction rather than by policy.

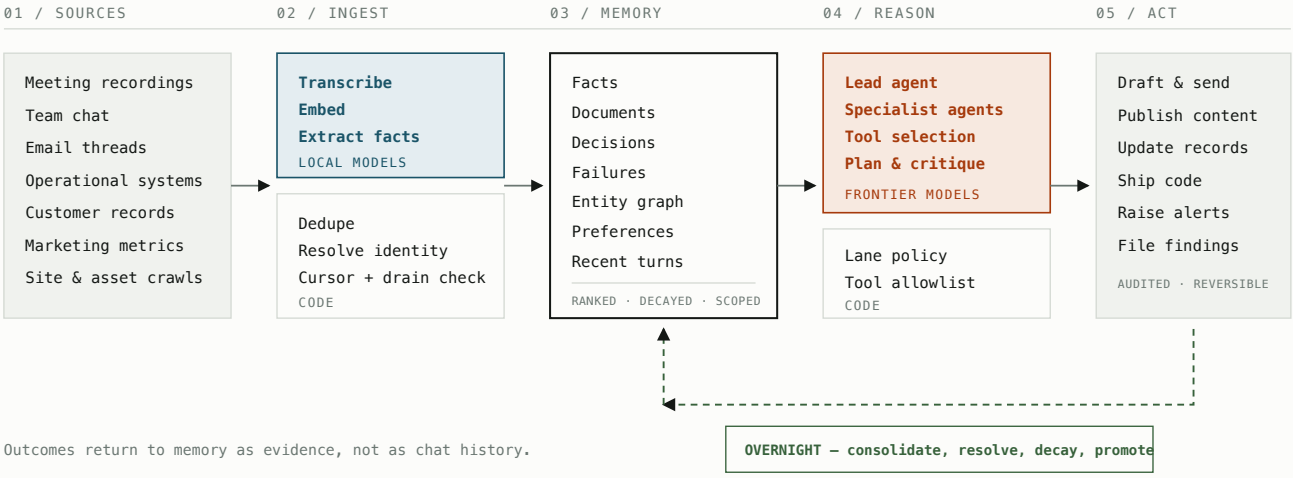
7	5.3k	1.9k	99.9%	0
retrieval layers per turn	durable facts in memory	autonomous jobs run	prompt input cached	metered memory cost

WHAT IT IS NOT

This is not a chatbot with a database attached, and not a workflow tool with a language model in one step. It is an always-on system with its own scheduler, its own long-term memory, its own governance layer over which tools it may touch, and a nightly pass in which it reviews and reconciles what it learned during the day. It runs continuously whether or not anyone is talking to it.

What happens between a meeting ending and an action being taken

The path runs left to right during the day and folds back on itself at night.



Everything in petrol runs on local hardware at no metered cost. Only the reasoning core in the fourth column is billed. The dashed return path is the reflective pass: it runs entirely on local models, so the system thinking about its own day costs nothing but electricity.

01 — INGEST

It reads everything, whether or not anyone asks it to

Every source the business already produces is pulled on a schedule and turned into durable, searchable records. Nobody pastes context into a chat window — by the time you ask, the system has already read the meeting.

LOCAL

Meetings, in full

Recordings are transcribed on-device, then indexed alongside the platform's own recap and action items. Both the summary and the raw transcript stay retrievable.

CODE

Cursors that can't lie

Each connector advances its bookmark only when it has provably drained the source. A truncated page logs a warning and re-reads next cycle instead of silently skipping.

CODE

One name per thing

A customer referred to five different ways across five systems resolves to a single identity, so history follows the subject rather than the spelling.

LOCAL

Facts, not transcripts

A local model reads each conversation and extracts standalone claims — who decided what, which constraint applies, what changed — with the source attached.

02 — REMEMBER

Memory is a ranked corpus, not a transcript log

Seven independent layers are consulted every turn and merged into one context budget. Relevance is scored, not assumed — and what stops being useful fades on its own.

LOCAL

Semantic search over everything

Vectors are generated on-premises, so the whole corpus is scored on every query rather than a sampled slice. Full-corpus scan: about five milliseconds.

CODE

Forgetting, on purpose

Importance decays on a curve unless a memory keeps earning retrieval. Pinned facts are exempt. Stale noise leaves the ranking without being deleted.

CODE

Duplicates reinforce, not multiply

Restating a known fact strengthens the existing record instead of writing a second copy that would otherwise flood retrieval with near-identical rows.

CODE

Failures are first-class

Things that went wrong are stored as their own retrievable class, weighted to recent history, so the system stops repeating a mistake without being told again.

03 — DECIDE

Work is queued, sequenced and governed

Autonomous jobs don't run as a free-for-all. They are claimed race-safely, held behind dependencies, and constrained to the tools their category of work actually needs.

CODE

Lanes and waves

Jobs declare a lane and their blockers. The executor enforces exclusivity within a lane and orders waves by dependency, so nothing races its own prerequisite.

CODE

Tool policy per lane

Each category of work gets an allowlist derived from what that lane actually used across thousands of prior runs — observed in shadow mode first, then enforced.

CLOUD

Plan, then critique

Significant work is planned, reviewed against known constraints and prior failures, and revised before anything executes.

CODE

A timeout on everything

Every scheduled handler runs under a hard backstop. A hung integration fails loudly on a clock rather than quietly occupying a worker forever.

04 — ACT

It has hands, and a record of what they did

LOCAL

Outbound drafting

First drafts are written by a local reasoning model against the memory of the account, then proofed by a cloud model before anything leaves the building.

CODE

Staleness gates

Automated sends are blocked when the underlying data is older than the action assumes. A stale queue parks itself instead of shipping confidently wrong messages.

05 — LEARN

Overnight, it goes back over its own day

LOCAL

Consolidation & synthesis

Facts about the same subject are merged into one authoritative statement, and related facts are lifted into higher-level insights the week implies but no single conversation contained.

LOCAL

Contradiction resolution

Pairs that are semantically close but not identical surface as possible conflicts and are resolved in favour of the most recent authoritative source.

The hardest bug class in an autonomous system is silent success

A job that throws gets noticed. A job that returns less than it should while reporting success can run for months. A deliberate hunt for that failure mode produced the changes below — each one a case where the system was reporting healthy and doing partial work. This is the difference between a demo and something a business can depend on.

WAS	NOW
Semantic search scored a fixed slice of the corpus, chosen by insertion order.	Every active memory is scored on every query. Measured cost of the change: three milliseconds.
Nightly consolidation grouped on a key that, by construction, could never have two members. It matched nothing — and reported success — for its entire life.	Grouping is subject-based. The first real pass merged dozens of duplicate records into authoritative facts.
The duplicate gate looked back thirty days — about two percent of the corpus. Any fact restated after a month wrote a fresh row.	The window covers everything the system has ever recorded. Restatement reinforces rather than duplicates.
An integration returned only its first page of results. Downstream summaries were built from an arbitrary subset.	Full pagination. The prompt receives the same number of items — selected from the whole set rather than a fraction of it.
A schedule expression the parser didn't understand fell back to running once a day, without complaint.	The full expression grammar is handled, and an unparseable schedule is an error rather than a default.
A job was marked healthy whenever its handler didn't throw — including handlers that caught every error internally.	Handlers report how much work they actually did. A job that runs cleanly and accomplishes nothing, repeatedly, raises itself as idle.

What it is built from

Deliberately boring infrastructure. The sophistication is in the memory model and the operating discipline, not in the parts list.

ON-PREMISES

- Single embedded database — write-ahead logging, streamed off-box continuously
- Local model runtime — embeddings, extraction, consolidation, drafting
- Supervised services — orchestrator, scheduler, executor, dashboard
- Private mesh networking — no public ingress

METERED

- Frontier reasoning — three tiers, matched to task difficulty
- Speech-to-text — batch transcription
- Media generation — on request only

CONNECTED SURFACES

- Meetings & internal chat
- Email & calendar
- Operational systems of record
- Customer relationship data
- Marketing, advertising & web infrastructure

OPERATING POSTURE

- Continuous off-box replication
- Documented cold-start recovery
- Structured audit on every write
- Self-reported job health

The architecture is portable. The hardware is an implementation detail.

The system described here runs on a single on-premises machine, but nothing in the design depends on that. The same architecture deploys onto existing server hardware or a private cloud tenancy without changing the model: sensitive data and the memory corpus stay inside the organisation's perimeter, and only the reasoning layer reaches out.

Figures reflect the system at the time of writing. This document describes architecture and operating principles; product names, client identifiers and internal metrics have been generalised for external distribution.