Inventory: Memory Plane

Memory Plane Inventory — AI Factory Audit

Date: 2026-05-09
Auditor: Chip Huyen (AgentForge)
Scope: Read-only probe. No mutations.
Task: Plan Task 1.1 — Memory Plane Inventory

---

1. Per-Store Table

Store	Endpoint / Path	Schema / Collections	Live Count	Write Path	Read Path	Owner Daemon	Status
mem0 / Qdrant	`http://localhost:9000` (mem0 API) / `http://localhost:6333` (Qdrant gRPC+HTTP)	5 collections: `mem0migrations` (0 pts), `sessions` (929 pts), `hivemind` (60,442 pts), `mem0_john` (865 pts), `knowledge` (31,274 pts)	**93,510 total vectors**	No caller found. mem0 API (`POST /add`) is NEVER called by any hook, tool, or daemon in `~/system/tools/` or `~/.claude/hooks/` . `hivemind.js` dual-writes to Qdrant `hivemind` collection directly via internal HTTP (port 6333).	No tool reads `localhost:9000` for queries. `hivemind.js semantic search` reads Qdrant `hivemind` collection directly via `qdrant-client` . `discover.js` does NOT query mem0.	`com.alai.mem0-server` (LaunchAgent, KeepAlive=true, PID 65706 alive, last exit was SIGTERM - 15)	HEALTHY (server alive, but ORPHANED — no producer writes to `mem0_john` or `knowledge` via the mem0 API)

Store	Endpoint / Path	Schema / Collections	Live Count	Write Path	Read Path	Owner Daemon	Status
Chroma	`~/.claude-mem/chroma/chroma.sqlite3`	1 collection: `cm__claude-mem`	**6,584 embeddings**	Unknown — no daemon or hook references `claude-mem` path in scanned tools. Likely written by a `claude-mem` MCP server or CLI tool directly.	Unknown — no caller found in `~/system/tools/` or `~/.claude/hooks/`.	None identified	PARTIAL (data exists, producer and consumer both untraced)
LightRAG	`http://localhost:9621`	Neo4J graph + NanoVectorDB + JsonKV storage; workspace `/app/data`	**999 processed docs, 1 failed** (pipeline_busy=true, 120 async locks pending — actively ingesting)	`~/.claude/hooks/lightrag-auto-ingest.sh` (PostToolUse: Write/Edit) — fires on writes to `~/.claude/projects/-Users-makinja/memory/*.md`, `~/system/specs/*.md`, and `/tmp/*-bookstack-*.md`. Also `com.alai.lightrag-outbox-ingest.plist` daemon.	`discover.js` — primary read path. Queries `https://lightrag.alai.no/query` (external hostname, not localhost). Fallback: if local hits < 3, LightRAG fallback fires.	`com.alai.lightrag-watchdog.plist`, `com.alai.lightrag-keepwarm.plist`, `com.alai.lightrag-backup.plist`, `com.john.lightrag-monitor.plist`, `com.alai.lightrag-migrate-pump.plist`	HEALTHY (serving, ingesting)
HiveDB (SQLite)	`~/system/agents/hivemind/hivemind.db`	7 tables: `agents` (139 rows), `memos` (100 rows), `intel` (17,551 rows), `subscriptions` (6 rows), `_litestream_seq`, `_litestream_lock`, `sqlite_sequence`	**17,551 intel rows** (NOTE: context memo said 64,889 — live probe shows 17,551; delta likely from live deletions or memo was stale)	`hivemind.js post <agent> <type> <message>` — agents call this CLI to write intel. Also dual-writes embeddings to Qdrant `hivemind` collection (best-effort, fire-and-forget).	`hivemind.js read/query/search` — text search + semantic search (cosine sim against local embeddings or Qdrant). `discover.js` does NOT query HiveDB directly.	`hivemind.js` (stateless CLI, no daemon; called ad-hoc by agents)	HEALTHY

Store	Endpoint / Path	Schema / Collections	Live Count	Write Path	Read Path	Owner Daemon	Status
.md auto-memory	`~/.claude/projects/-Users-makinja/memory/`	123 `.md` files (MEMORY.md index + per-topic files + feedback memos + _archive/)	**123 files**	**Claude Code's built-in auto-memory system** (native Claude Code feature — writes `.md` files after conversations automatically, not via any explicit hook or daemon). `lightrag-auto-ingest.sh` PostToolUse hook then ingests these into LightRAG when they are written/edited.	CLAUDE.md "Context Loading" section instructs John to `Read` specific files directly. `discover.js memory "<topic>"` is documented as LightRAG-backed (reads LightRAG, not the .md files directly).	Built-in Claude Code (no external daemon)	HEALTHY (write path functional; read path partially bypassed — LightRAG index only 999 docs, not all 123 .md files confirmed ingested)

2. Producer ? Consumer Matrix

Producer	Store Written	Consumer	Notes
Claude Code built-in auto-memory	`~/.claude/projects/-Users-makinja/memory/*.md` (123 files)	`lightrag-auto-ingest.sh` hook (secondary producer → LightRAG)	Auto-memory is Claude Code native. The .md write triggers the hook.
`lightrag-auto-ingest.sh` (PostToolUse hook)	LightRAG `http://localhost:9621`	`discover.js` (primary RAG consumer)	Only fires on Write/Edit tool calls to in-scope paths. Does NOT write to mem0.
`com.alai.lightrag-outbox-ingest.plist` daemon	LightRAG	`discover.js`	Batch ingest pipeline for outbox staging

Producer	Store Written	Consumer	Notes
`hivemind.js post` (called by agent tools)	HiveDB SQLite `hivemind.db` + Qdrant `hivemind` collection (dual-write)	`hivemind.js read/query/search` (CLI)	Qdrant `hivemind` = 60,442 vectors; SQLite intel = 17,551 rows — divergence suggests Qdrant has historical vectors beyond current SQLite rows (possibly from bulk migration)
NOBODY	mem0 API (`localhost:9000/add`) — `mem0_john` collection (865 pts), `knowledge` collection (31,274 pts)	**NOBODY** reads via mem0 API either	**WIRE BREAK: mem0_john has 865 facts that were presumably written at some point (possibly during initial mem0 setup / manual population), but no current tool, hook, daemon, or agent calls `POST localhost:9000`. The mem0 API is a running server with no active clients.**
NOBODY identified	Chroma `~/.claude-mem/chroma/` (6,584 embeddings)	**NOBODY identified**	Chroma has data (6,584 embeddings in `cm__claude-mem`) but producer and consumer are both untraced in current tooling. Likely written by a `claude-mem` MCP tool in a previous iteration.
`com.john.session-archiver.plist`	Likely `sessions` Qdrant collection (929 pts)	`discover.js --sessions` (reads sessions SQLite, not Qdrant)	Sessions exist in Qdrant but `discover.js` reads from a local SQLite `sessions` table, not via mem0 or Qdrant API
`rag-router.js learn`	`~/system/databases/flywheel.db` (SQLite: interactions + rag_cache)	`rag-router.js query` (cache-hit path)	Sixth store — flywheel SQLite, not listed in original inventory. Routes: cache → local Ollama → external. Does not touch mem0.

---

3. SoR Gap Analysis — Duplicated Fact Classes

Fact Class	Stores Containing It	Designated SoR	Derivative / Shadow	Gap / Conflict
Agent intel / decisions	HiveDB `intel` table (17,551 rows) + Qdrant `hivemind` collection (60,442 vectors)	HiveDB SQLite (primary; `hivemind.js` writes here first)	Qdrant `hivemind` (dual-write, best-effort)	60,442 Qdrant vectors vs 17,551 SQLite rows = **3.4x divergence**. Qdrant likely contains orphaned vectors from deleted/purged SQLite rows, or a bulk historical migration that wasn't reflected in SQLite. No reconciliation daemon exists.
Session summaries / history	Qdrant `sessions` (929 pts) + likely local session SQLite (referenced by `discover.js`) + `.md` memory files (MEMORY.md index)	Undefined — no explicit SoR designation	All three are partial	`discover.js --sessions` reads SQLite, not Qdrant `sessions`. Who writes Qdrant `sessions`? Untraced.
John's personal facts / preferences	mem0 `mem0_john` collection (865 vectors) + `.md` auto-memory files (123 files) + LightRAG (999 docs, subset overlapping .md files)	Intended SoR: mem0 (`mem0_john`) — but NO active writer. Actual SoR: `.md` files (Claude Code writes here).	LightRAG is downstream derivative of .md files via `lightrag-auto-ingest.sh`	**Critical SoR conflict**: 865 facts in mem0 are STALE (last written at setup, no ongoing writes). 123 .md files are current. LightRAG is a partial index of .md files. Three stores claim the same fact class with no reconciliation.
Knowledge base / operational docs	mem0 `knowledge` collection (31,274 vectors) + LightRAG (999 docs, BookStack exports) + Chroma (6,584 embeddings)	Undefined	All three parallel	`knowledge` collection in mem0 has 31,274 vectors — largest in mem0, but again no active writer via mem0 API. Origin unknown. Chroma `cm__claude-mem` (6,584) is also an orphan with no identified current writer or reader.

Fact Class	Stores Containing It	Designated SoR	Derivative / Shadow	Gap / Conflict
HiveMind broadcast intel	HiveDB `hivemind` Qdrant collection (60,442) + HiveDB SQLite `intel` (17,551)	HiveDB SQLite is the write authority	Qdrant `hivemind` is derivative (dual-write from `hivemind.js`)	No `hivemind` HTTP API exists (confirmed: port 3001 is Drop API). Qdrant `hivemind` is only queryable via `hivemind.js` semantic search CLI, not accessible to other tools.

---

4. Critical: The .md vs mem0 Wire Break

What was supposed to happen

The architecture assumes mem0 (`http://localhost:9000`) is the structured personal memory SoR for John. The `mem0_john` collection exists with 865 facts. The `sessions` collection has 929 entries. The server is alive and healthy.

What actually happens

Step 1 — .md files are written by Claude Code natively.
Claude Code has a built-in auto-memory feature that writes conversation summaries and facts as `.md` files into `~/.claude/projects/-Users-makinja/memory/`. This is NOT a hook or daemon — it is a built-in Claude Code behavior. No line of code in `~/system/` controls this write.

Step 2 — lightrag-auto-ingest.sh hooks into the .md write.
File: `~/.claude/hooks/lightrag-auto-ingest.sh` (PostToolUse on Write/Edit).
This hook detects when a `.md` file is written to `~/.claude/projects/-Users-makinja/memory/*.md` and fires a background `curl POST` to LightRAG (`http://localhost:9621/documents/text`). This is the ONLY downstream pipeline from .md files.

Step 3 — mem0 API is never called.
Grep across all of:

- `~/system/tools/*.js` — 0 files call `localhost:9000`
- `~/.claude/hooks/*.sh` — 0 files call `localhost:9000`
- `~/system/daemons/` — not scanned exhaustively but mem0-server plist confirms it's only a server, not a writer

- `pi-orchestrator.js` — the one hit for `localhost:9000` is SonarQube (port 9000 collision), not mem0

The exact wire break: There is no `POST http://localhost:9000/add` call anywhere in the active system. The mem0 server was built and populated (865 facts in `mem0_john`, 31,274 in `knowledge`) at some point — likely during initial setup or a one-time migration — but the "auto-write to mem0" integration was never wired into the live pipeline. The `lightrag-auto-ingest.sh` hook was written instead, routing .md → LightRAG, leaving mem0 as a read-only relic with stale data.

CEO complaint root cause confirmed: "implementation is not ideal — memory writes to .md files instead of mem0" is accurate. The intended SoR (mem0) has no active producer. The actual write path is: `Claude Code → .md files → lightrag-auto-ingest.sh → LightRAG`. mem0 is running, healthy, and populated with 865+31,274 stale vectors that nobody reads.

HiveDB relationship

HiveDB (`hivemind.db`) is a SEPARATE concern from personal memory. It is the agent broadcast / intel bus, not John's fact store. However, the Qdrant `hivemind` collection (60,442 vectors) lives in the same Qdrant instance as `mem0_john`, creating the appearance of a unified store when it is actually two separate logical systems sharing infrastructure.

---

5. Store Status Summary

Store	Healthy?	Active Producer?	Active Consumer?	Data Fresh?
mem0 / Qdrant `mem0_john`	Yes	NO	NO	NO — 865 facts, stale
mem0 / Qdrant `knowledge`	Yes	NO	NO	NO — 31,274 vectors, stale
mem0 / Qdrant `sessions`	Yes	Unknown	NO	Unknown
mem0 / Qdrant `hivemind`	Yes	Yes (hivemind.js dual-write)	Yes (hivemind.js semantic search)	YES
HiveDB SQLite	Yes	Yes (hivemind.js CLI)	Yes (hivemind.js CLI)	YES — 17,551 rows
LightRAG	Yes	Yes (lightrag-auto-ingest.sh hook + outbox daemon)	Yes (discover.js)	YES — 999 docs, pipeline busy
Chroma	Yes (file exists)	UNKNOWN	UNKNOWN	Unknown origin
.md auto-memory	Yes	Yes (Claude Code native)	Partial (direct Read + LightRAG index)	YES — 123 files
Flywheel SQLite	Presumed yes	Yes (rag-router.js learn)	Yes (rag-router.js query)	Unknown

Open Questions

1. **Chroma write/read path**: Who wrote 6,584 embeddings to `~/.claude-mem/chroma/cm__claude-mem`? Which tool or MCP server reads from it? The `claude-mem` MCP is referenced in settings but not found in scanned tool code. Needs: `grep -r "claude-mem\|chroma" ~/.claude/settings.json` and MCP server registry audit.
2. **Qdrant `sessions` writer**: Who writes 929 session vectors to the `sessions` Qdrant collection? `com.john.session-archiver.plist` is a candidate but the script path was not read. Needs: `cat ~/Library/LaunchAgents/com.john.session-archiver.plist` + script inspection.
3. **Qdrant `knowledge` origin**: 31,274 vectors in `knowledge` — when were they written and from what source? No active writer found. Possible: one-time BookStack bulk ingest or a migration. Check `~/system/mem0/server.py` for any bulk-load routines at startup.
4. **HiveDB vector divergence**: 60,442 Qdrant vectors vs 17,551 SQLite intel rows. Are the extra ~43K vectors orphaned (deleted SQLite rows without Qdrant cleanup), or does Qdrant have independent content? Needs: sample Qdrant payload IDs vs SQLite `id` column cross-check.
5. **LightRAG external hostname**: `discover.js` queries `https://lightrag.alai.no/query` (external URL from config), not `http://localhost:9621`. Is there a Caddy/Cloudflare proxy routing `lightrag.alai.no` → `localhost:9621`? If that proxy is down, `discover.js` would silently fail to read from LightRAG despite the local container being healthy.
6. **mem0_john 865 facts provenance**: When were these written? Is there a one-time ingestion script (e.g., `~/system/mem0/populate.py` or similar)? If the facts are high-quality (personal preferences, CEO directives), they are the most actionable store to re-wire as the active SoR.
7. **`rag-router.js` flywheel.db size and health**: Not probed live. Needs `sqlite3 ~/system/databases/flywheel.db "SELECT count(*) FROM interactions; SELECT count(*) FROM rag_cache;"`.
8. **mem0 `server.py` — does it expose `/add` or `/search` routes?**: Confirmed health endpoint works. Need to verify actual API surface to confirm if a PostToolUse hook calling `POST localhost:9000/add` would work as-is without code changes to mem0.

Revision #2
Created 2026-05-09 19:44:18 UTC by John
Updated 2026-06-14 20:02:53 UTC by John