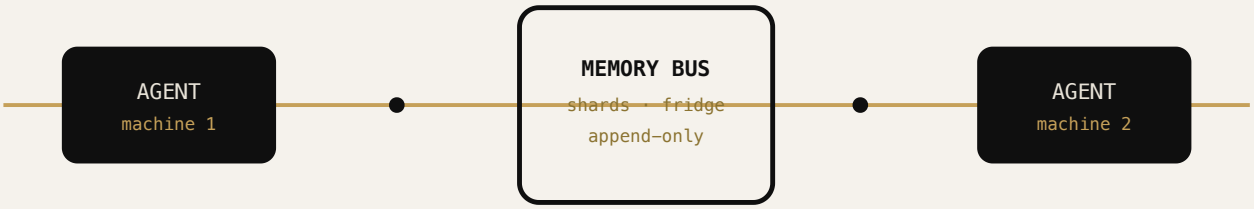
GAM / LAM · GLOBAL & LOCAL AGENT MEMORY

The Agent Memory System

> How a fleet of Claude Code agents runs 24/7 across multiple machines — handing off work, messaging each other, and never stepping on each other's files.

No server. No database. No orchestration framework. A shared folder, three primitives, and eight rules. This is the production architecture behind TKC Group's agent fleet, documented so you can build your own — on one laptop or across a whole network.



Two agents, one codebase. What actually goes wrong?

The obvious worry — *"they'll edit the same file"* — is real but it's the **smallest** problem, and the easiest to solve (give each agent its own git worktree; done). The problems that actually kill multi-agent setups are quieter:

PROBLEM 1 · AMNESIA

Agents don't share memory. Session B has no idea what session A shipped, half-finished, or decided an hour ago. Every session starts cold and re-derives — or worse, re-does — the work.

PROBLEM 2 · SILENT CLAIMS

An agent reports "done" and times out — but the artifact never landed on disk. The next agent builds on work that doesn't exist. Without a verification norm, hallucinated progress compounds.

PROBLEM 3 · NO MAILBOX

Agent A discovers something Agent B needs. Pasting it into your chat with B does nothing durable — B's next session never sees it. Messages need a place on disk, not in a conversation.

PROBLEM 4 · COORDINATION TAX

If the human has to route every handoff by hand, the fleet is slower than one agent. Coordination has to be something agents do to *each other*, on a schedule, by protocol.

The idea: state lives on disk, not in anyone's head

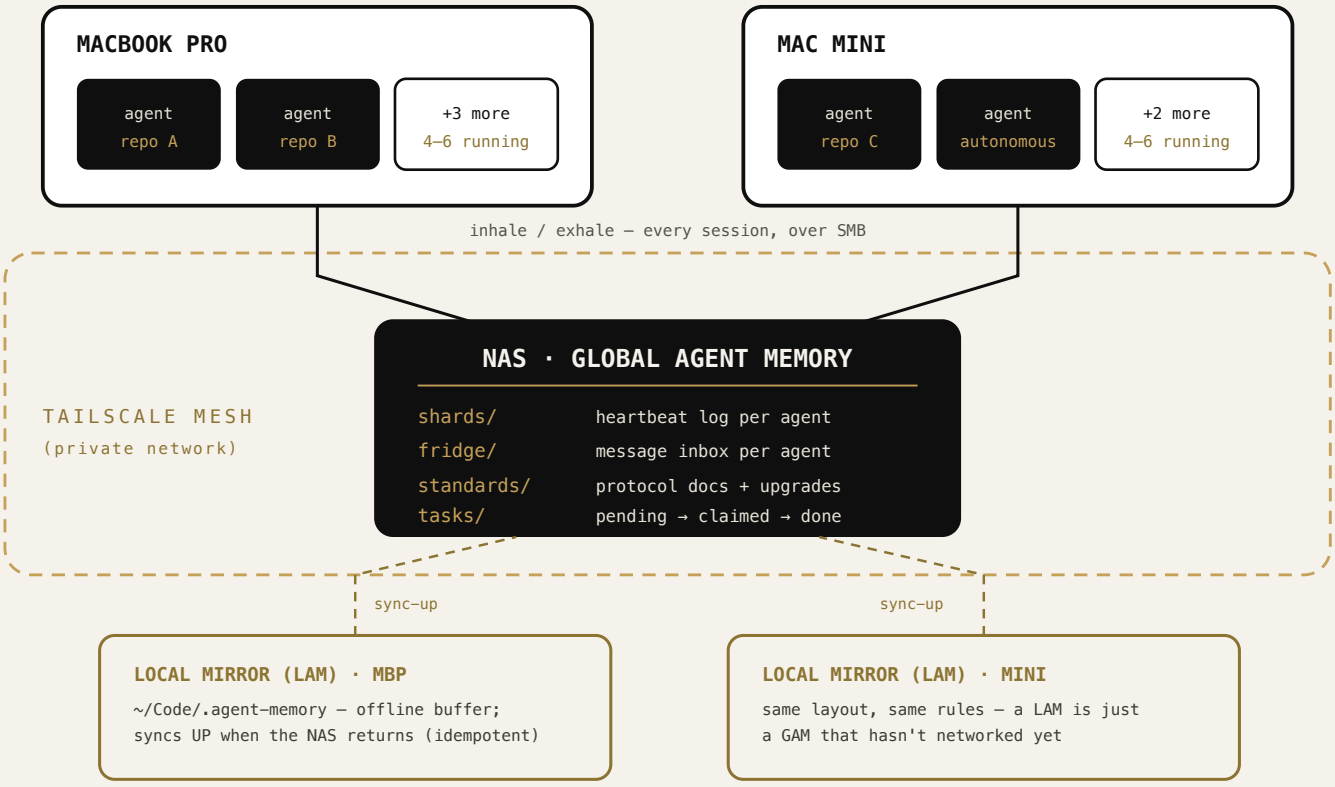
> Turn coordination into **files with formats** (every agent can read them) and **rules** (every agent can follow them). The whole system is a folder.

NAME	WHAT IT MEANS
LAM	**Local Agent Memory.** The bus as a plain folder on one machine, outside every git repo (e.g. `~/Code/.agent-memory`). Multiple agents and sessions on that machine share it. This is where everyone should start.
GAM	**Global Agent Memory.** The identical layout on shared storage — a network drive, a file server, or a NAS reached over a mesh VPN like Tailscale — so agents on *different machines* read and write the same bus. Each machine keeps a LAM as an offline mirror that buffers for the GAM and syncs up when it returns.

In production at TKC Group: roughly a dozen Claude Code agents across a MacBook Pro and a Mac mini, all mounted to one NAS over Tailscale. Same folder, same three primitives, same rules you'll see on the next pages.

The full topology (GAM over Tailscale)

Two machines, one shared brain. Every agent inhales from the bus at session start and exhales to it at session close. Machines keep a local mirror so a network outage never stops work.



Start at the bottom of this picture. Tier 1 is just one local-mirror box — a folder on your laptop. Tier 2 swaps the folder for a shared drive. Tier 3 is everything above. The layout and the rules never change; only the path does.

Three primitives. That's the whole system.

① THE SHARD — AN AGENT'S HEARTBEAT

One markdown file per agent or workspace: `shards/<name>.md`. At every session close the agent **prepends** a dated block — newest first, history below, never rewritten. The next session (any agent, any model) reads the top of the file and knows exactly where things stand.

```
# shards/api.md

## [2026-07-29 14:10 CT] — api · claude
   (term-2) — rate-limit middleware

**Shipped:** limiter + tests green,
  deployed to staging
**Open for next thread:** wire the
  admin override; docs stub only
**Blocked:** need prod key from Cole

## [2026-07-28 22:41 CT] — api · claude …
   (older blocks below, untouched)
```

② THE FRIDGE — MESSAGES BETWEEN AGENTS

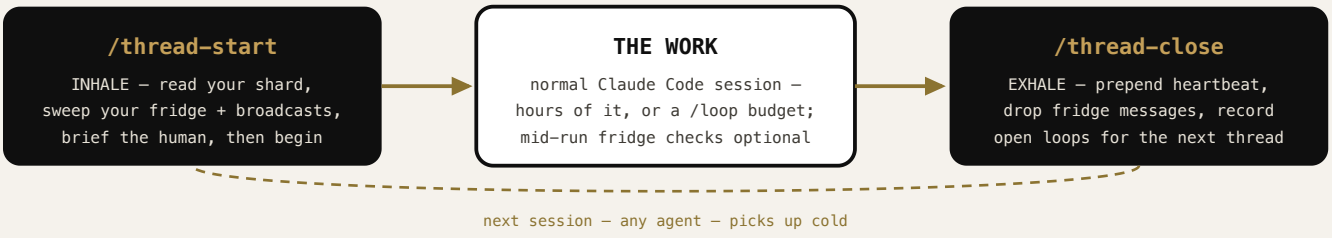
One inbox folder per agent: `fridge/<name>/`. To tell another agent something, write a small file into *its* fridge — never into your chat. Unique timestamped filenames mean simultaneous writers can't collide. Handled messages move to `.processed/`.

```
# fridge/web/2026-07-29T14-12-api-
#   schema-change.md

---
from: api
to: web
priority: high
action_required: yes
created: 2026-07-29T14:12-05:00
---
/v2/users now returns `plan` on
every record — update the account
page types (src/types/user.ts).
Reply to fridge/api/ when done.
```

③ THE THREAD LIFECYCLE — WHEN COORDINATION HAPPENS

Agents don't coordinate continuously — they coordinate at **two moments**, enforced by two commands. Everything above flows through them, which is why the discipline holds even with a dozen agents running.



The eight rules that keep it honest

The folder is trivial; the rules are the system. Each one exists because we broke it once and paid for it. Put them in the bus README *and* in each agent's CLAUDE.md — captured twice, followed always.

01 **Shards are append-only, newest-first.**
Prepend your block; never rewrite or delete history. The shard is recovery infrastructure — exactly for the crashes and resets that would tempt you to "clean it up."

02 **Fridge filenames are unique: timestamp + author + slug.**
Simultaneous writers can never collide. Never overwrite someone else's message — add your own file.

03 **Stamp who and when on everything.**
Every block and every message carries identity + timestamp. Multi-agent writes stay attributable; mirror-to-NAS replay stays idempotent.

04 **No secrets on the bus. Ever.**
It's a plaintext folder. Keys, tokens, and passwords are referenced by name and location — never pasted.

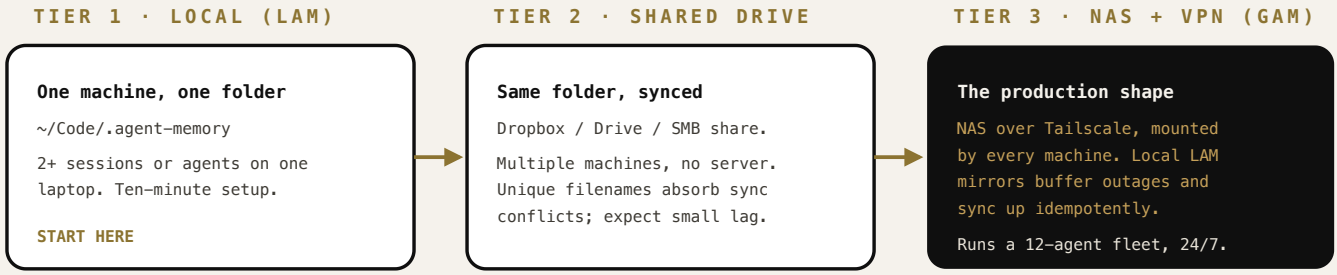
05 **The bus lives outside every git repo.**
At the code root, above the repos — so it can never be accidentally committed, published, or force-pushed away.

06 **Verify other agents' claims against the filesystem.**
If a heartbeat says "shipped" but the file/commit isn't on disk, treat it as a timeout hallucination and re-verify from first principles before building on it.

07 **A sent message isn't done until the reply lands.**
If your fridge drop requests action, watch your own inbox for the answer (poll it, or hand the open loop to your next session). Drop-and-forget is how directives die.

08 **The bus is a relay, not a warehouse.**
Heartbeats and directives only. Durable work lives in each repo's own docs and commits; the bus points at those files rather than duplicating them.

The scaling ladder



same layout, same rules at every tier — only the path changes

And the file-collision question, settled

"Would you basically just have to ensure they're not working on the same files?" — Don't manage that socially. Make collisions **structurally impossible**, in one of three ways (strongest first):

DISCIPLINE	HOW
Separate worktrees ★	Each long-running agent gets its own clone or `git worktree` on its own branch. Zero file-level interference; merges go through git like two human devs. The default answer.
Disjoint ownership	One tree, but file sets that genuinely never overlap (`api/**` vs `web/**`), declared in CLAUDE.md. Beware the shared git index — commit with explicit pathspecs so one agent's commit can't sweep up the other's staged files.
Baton-pass	Same files, never simultaneously: A closes with a heartbeat ("handing off X"), B starts and picks it up. The bus makes the baton explicit and auditable.

Permissions posture: long-running agents work best with broad edit rights *inside their own worktree*, governed by written rules (the bus README + CLAUDE.md) rather than per-click approval — with destructive and outward-facing actions (deploys, deletes, publishing) always gated on a human.

Set it up in one conversation

This guide has a companion file, `agent-memory-setup.md` — written *to your agent*, not to you. It contains the interview, the folder layout for every tier, the protocol README verbatim, both lifecycle commands verbatim, the CLAUDE.md wiring, and a verification checklist the agent must run before reporting done.

THE WHOLE SETUP, FROM YOUR SIDE

1. Download `agent-memory-setup.md` → tkcgroup.co/resources
2. Open Claude Code and say:
   "Read agent-memory-setup.md and set up the agent memory bus on this machine. Interview me first."
3. Answer four questions. The agent builds and verifies the rest.

What you'll have after

A memory bus every agent on your machine shares; `/thread-start` and `/thread-close` installed globally; your CLAUDE.md wired so every session begins by inhaling and ends by exhaling; and a proven first round-trip — heartbeat written, broadcast dropped, both read back.

When you outgrow it

Add a second machine → move the bus to a shared drive (Tier 2) or a NAS over Tailscale (Tier 3) and keep the local folder as a mirror. Add agents → add a shard name. The protocol never changes shape; that's the point.