

Jamals geheugenlandschap – hoe het echt in elkaar zit

In het kort

Je hebt niet één **geheugen**, maar een **stapel van geheugens**.

De kern is:

- 1. **Gedeeld team-geheugen** voor blijvende feiten tussen bots.
- 2. **Lokaal file-geheugen** voor diepte, handoffs en dossiers.
- 3. **Telegram-archief** voor exacte gespreksgeschiedenis.
- 4. **Verslagen** voor leesbare kennis en overdrachten.
- 5. **Skills + hooks + settings** voor gedrag, regels en automatisering.
- 6. **Sessielogs** voor forensiek en debuggen.

Dus: een bot “onthoudt” niet vanzelf alles in zijn model.

Een bot weet iets opnieuw omdat het uit **deze lagen** kan lezen.

1. De hoofdkaart

LAAG	FUNCTIE	HOOFDPAD / ROUTE	GEDEELTE
Shared memory	Korte blijvende teamfeiten	python3 ~/.claude/scripts/shared_memory_client.py recall/remember	Ja
File memory	Rijke kennis, facts, handoffs, projectnotities	~/.claude/projects/-Users-macbookjamal/memory/	Gedeelte voor bo op deze Mac / vi

LAAG	FUNCTIE	HOOFDPAD / ROUTE	GEDEELTE
			repo-kopieën
Telegram-archief	Exacte chatgeschiedenis met Jamal	~/ .claude/sync-in/telegram-archive/	Praktisch gedeeld als bron
Hermes-verslagen	Markdown-bron van verslag-kennis	/Users/macbookjamal/hermes-data/verslagen/	Leesbron voor Hermes, ook teamker
Scripts	Bruggen naar geheugen, Telegram, queue, checks	~/ .claude/scripts/	Ja, function gedeeld
Skills	Instructies per taak of domein	~/ .agents/skills/ en ~/ .codex/skills/	Ja
Kimi lokaal geheugen	Kimi's eigen handoffs en kleine lokale kennislaag	~/kimi-workspace/geheugen/	Nee, vooral Kimi-lok
Claude projectlogs	Sessies, toolresultaten, forensiek	~/ .claude/projects/	Nee, vooral technisch bewijs
Codex sessies	Codex-eigen sessiegeschiedenis	~/ .codex/sessions/	Nee, per runner

2. Wat is het echte gedeelde geheugen?

De meest letterlijke gedeelde laag is deze route:

- **Opzoeken:** `python3 ~/.claude/scripts/shared_memory_client.py recall "<vraag>"`
- **Opslaan:** `python3 ~/.claude/scripts/shared_memory_client.py remember "<feit>"`
hermes

Dat staat ook letterlijk zo in je actieve teaminstructies:

- “The Haarvisie team shares one memory”
- “you can read and write it from your shell”

Wat dit geheugen **wel** is:

- een plek voor **blijvende teamfeiten**
- beslissingen
- recepten
- werkende fixes
- projectwaarheden die ook later nog nuttig zijn

Wat dit geheugen **niet** is:

- geen volledige chatdump
- geen goede plek voor wachtwoorden
- geen plek voor ruis of halve tussenstappen

Belangrijke les uit je eigen geheugenhistorie:

- als een feit **nooit is opgeslagen**, kan recall het ook nooit terughalen
- veel “geheugenverlies” blijkt in de praktijk **schrijfdiscipline-probleem**, niet een kapotte zoekfunctie

3. Het rijke file-geheugen

Naast shared memory heb je een veel diepere laag:

- `~/.claude/projects/-Users-macbookjamal/memory/`

In je instructies staat letterlijk:

- “There is also a richer file memory at `~/ .claude/projects/ -Users - macbookjamal/memory/` ”

Dit is in de praktijk je:

- handoff-geheugen
- projectgeheugen
- referentie-geheugen
- les-/feedback-geheugen
- open-punten- en besluitgeheugen

Dit is de laag waar vaak de **volledige context** zit, terwijl shared memory meer de **kernfeitjes** draagt.

Kort:

- **shared memory** = kort en overdraagbaar
 - **file memory** = rijk en diep
-

4. Telegram is je waarheid voor “wat is er echt gezegd?”

Voor alles wat te maken heeft met:

- wat Jamal zei
- wanneer hij iets zei
- welke nuance hij gebruikte
- wat er in een eerdere Telegram-run stond

is deze laag leidend:

- `~/ .claude/sync-in/telegram-archive/YYYY-MM-DD.md`

Instructieregel:

- “Do not pretend the whole Telegram history is already in context”
- “read the local Telegram archive before answering”

Dus een bot hoort bij Telegram-vragen **niet te gokken uit het hoofd**.

Hij hoort lokaal te zoeken in het archief.

Dat archief is dus geen “bijzaak”.

Het is een **primaire bron**.

5. Verslagen zijn een aparte kennislaag

Voor eerdere onderzoeken, analyses en overdrachten gebruik jij ook:

- /Users/macbookjamal/hermes-data/verslagen

In je Hermes-instructies staat letterlijk dat daar **eerst lokaal** gezocht moet worden bij vragen over:

- verslagen
- onderzoeken
- beslissingen
- acties
- botgeschiedenis

Dit is dus niet hetzelfde als shared memory.

Verslagen zijn:

- langer
- leesbaarder
- vaak beter als dossier
- beter voor publicatie en teruglezen op je telefoon

Shared memory is kort.

Verslagen zijn breed.

6. Skills zijn geen geheugen, maar wel gedrag

Je vroeg ook waar skills vandaan komen.

Die komen hier vandaan:

- `~/.agents/skills/`
- `~/.codex/skills/`
- plus plugin-caches onder `~/.claude/plugins/cache/...` of `~/.codex/plugins/cache/...`

Een skill is meestal:

- een `SKILL.md`
- met instructies
- soms met scripts of references erbij

Een skill is dus **geen opgeslagen feit** over jouw bedrijf.

Een skill is eerder:

- een werkwijze
- een checklist
- een taak-recept
- een domeininstructie

Voorbeeld:

- `conductor-werkwijze` bepaalt **hoe** de bot moet meten en beslissen
- `haarvisie-shared-memory` bepaalt **hoe** shared memory gebruikt moet worden
- `haarvisie-report-writing` bepaalt **hoe** een verslag moet worden geschreven en gepubliceerd

Kort:

- **geheugen** = wat waar is
- **skill** = hoe je werkt

7. Hooks en settings zijn de onzichtbare automatisering

Een belangrijk stuk van je systeem zit niet in het model, maar in hooks en settings.

Gemeten Stop-hooks op 8-8-2026:

- `~/ .claude/settings.json`
- `~/ .codex/hooks.json`

Daar hangen lokale scripts aan zoals:

- `telegram-reply-enforcer.py`
- `stop-checkpoint.py`
- `auto-obsidian-stop.py`
- `slack-mijlpaal-reminder.py` (in `.claude/settings.json`)

Die hooks doen dingen als:

- controleren of er via Telegram is geantwoord
- sessie-activiteit wegschrijven
- lichte housekeeping
- herinneren aan Slack-mijlpaallogs

Dit is belangrijk omdat een deel van jouw “geheugenbeleving” eigenlijk uit **automatische bijschrijvers** komt.

Dus niet:

- “de bot onthoudt dat vanzelf”

maar:

- “een hook/script heeft het op het juiste moment vastgelegd”

8. Sessielogs zijn technisch bewijs, geen nette kennislaag

Je hebt ook zware technische sporen in:

- `~/ .claude/projects/`
- `~/ .codex/sessions/`

Daarin zitten:

- sessiegeschiedenis
- tool outputs
- hook summaries
- debugsporen
- oude diagnosepaden

Deze laag is goud voor:

- forensiek
- fouten terugzoeken
- bewijzen wat er echt draaide

Maar slecht voor:

- snel menselijk teruglezen
- nette kennisopslag

Dus:

- **goed als bewijs**
- **slecht als hoofdgeheugen**

9. Kimi, Hermes en de andere bots hebben ook lokale stukken

Niet elke bot leeft alleen uit het gedeelde teamgeheugen.

Sommige bots hebben een eigen lokale laag.

Voorbeeld Kimi:

- `~/kimi-workspace/geheugen/`
- `~/kimi-workspace/.kimi-handoff.md`

Dat is handig voor Kimi zelf, maar het is **niet automatisch teambreed**.

Daarom is je teamregel terecht streng:

- waardevolle feiten moeten uiteindelijk naar **shared memory**
- anders blijven ze hangen in een lokale hoek waar de rest ze niet terugvindt

Dit is precies waarom er eerder verwarring ontstond tussen:

- “het is ooit getest”
 - en
 - “niemand kan het terugvinden”
-

10. Wat onthoud ik zelf, als Hermes?

Eerlijk antwoord:

- **tijdelijk** onthoud ik dingen in de lopende sessie
- **blijvend** onthoud ik alleen wat in jouw systemen is opgeslagen

Dus ik ben niet zelf het langetermijngeheugen.

Ik ben een werker met toegang tot:

- shared memory
- file memory
- Telegram-archief
- verslagen
- skills
- sessielogs

Mijn “geheugen” over sessies heen is dus in feite:

- wat ik opnieuw kan **inladen**

Niet:

- wat mijn model vanzelf eeuwig bewaart
-

11. Wat is nu leidend bij welk soort vraag?

Gebruik deze simpele besliskaart:

VRAAGTYPE	EERSTE BRON	TWEEDE BRON	OPMERKING
Wat hebben we besloten?	shared memory	file memory	Teamfeit eerst
Wat zei Jamal letterlijk?	Telegram-archief	sessielogs	Chat is bron
Is dit eerder onderzocht?	verslagen	file memory	Dossier eerst
Hoe moet een bot werken?	skill / AGENTS.md	settings / hooks	Gedrag, geen feit
Waarom ging iets technisch mis?	sessielogs / hook output	scripts + settings	Debuglaag
Wat geldt teambreed?	AGENTS.md + teamregels	shared memory	Contractlaag

12. Mijn eerlijke conclusie

Je systeem is al behoorlijk rijk.

Het probleem is niet dat je **geen geheugen** hebt.

Het probleem is eerder dat je **meerdere soorten geheugen** hebt die door elkaar lopen:

- feiten
- dossiers
- chats
- skills
- hooks
- logs

Als iemand dat niet uit elkaar houdt, voelt het alsof bots willekeurig iets wel of niet weten.

De werkelijkheid is:

- iets kan **bestaan** in een verslag
- maar **niet** in shared memory
- of in Telegram staan
- maar niet in de facts-laag
- of als lokale Kimi-notitie bestaan
- maar niet teambreed terugvindbaar zijn

Dat is de echte bron van veel verwarring.

13. Mijn advies

Houd voortaan dit model aan:

1. **Shared memory** voor blijvende teamfeiten.
2. **File memory** voor rijke context en handoffs.
3. **Telegram-archief** voor letterlijke waarheid van gesprekken.
4. **Verslagen** voor menselijk leesbare kennis.
5. **Skills** voor gedrag.
6. **Hooks/logs** voor automatisering en bewijs.

De fout die bots vaak maken is dat ze deze lagen door elkaar behandelen alsof het allemaal “geheugen” is.

Dat is technisch niet waar.

Het zijn **verschillende geheugens met verschillende taken**.

14. Wat ik hiervoor echt heb gemeten

Gemeten op **zaterdag 8 augustus 2026**:

- `~/.claude/projects/-Users-macbookjamal/memory/` = **2914 bestanden**
- `~/.claude/sync-in/telegram-archive/` = **195 bestanden**
- `~/.claude/scripts/` = **362 bestanden**
- `/Users/macbookjamal/hermes-data/verslagen/` = **421 bestanden**
- `~/.agents/skills/` = **15144 bestanden**
- `~/.codex/skills/` = **56 bestanden**
- `~/kimi-workspace/geheugen/` = **10 bestanden**
- `~/.claude/projects/` = **8578 bestanden**
- `~/.codex/sessions/` = **167 bestanden**

Ook geverifieerd:

- actieve teaminstructie noemt expliciet het `shared-memory-commando`
- actieve Hermes-instructie noemt expliciet Telegram-archief en `hermes-data/verslagen`
- actieve Stop-hooks staan in `~/.claude/settings.json` en `~/.codex/hooks.json`

Let op, eerlijk:

Ik heb in deze beurt **niet live in gBrain zelf gezocht**.

Ik heb wel gemeten dat gBrain in jouw teamregels als belangrijke zoeklaag genoemd wordt.

Dus in dit verslag behandel ik gBrain als **bedoelde / beleidslaag**, niet als live door mij geverifieerde primaire runtime-bron van deze beurt.