 LUISTER DIT VERSLAG

-10% 1,0x +10% +25% 1,0x

⬇ Download naar telefoon

▶ Lees mee met de podcast

---

1. Het probleem in gewone taal

Een bot heeft geen geheugen zoals jij.

Hij heeft een **werkgeheugen**, het zogeheten context-venster. Zie het als een tafel waar alles op moet passen: jouw opdracht, de bestanden die hij las, de uitkomsten van zijn commando's, zijn eigen antwoorden.

Die tafel is groot, maar niet oneindig.

Raakt de tafel vol, dan gebeurt er een **compact**. De bot maakt dan een korte samenvatting van alles wat er lag, veegt de tafel leeg en legt alleen die samenvatting terug. Dat is geen kopie. Dat is een navertelling.

En daar gaat het mis. In zo'n samenvatting sneuvelt bijna altijd hetzelfde soort informatie: de precieze woorden van de oorspronkelijke opdracht, en waarom hij ergens mee bezig was.

Er is nog een tweede oorzaak, en die is verrassender. Uit onderzoek van Chroma (18 modellen getest) blijkt dat een model een lange tafel **niet gelijkmatig leest**. Wat in het midden ligt, ziet hij het slechtst. Klassiek onderzoek noemt dat "lost in the middle".

Gevolg: een opdracht die bovenaan een volgelopen sessie stond, is de bot vaak al kwijt **vóór** de compact. De compact krijgt de schuld, maar is niet de enige dader.

Bron context rot: https://www.trychroma.com/research/context-rot

En dan is er nog een derde ding, dat bij jou het zwaarst weegt.

Elke opdracht uit de wachtrij start een **compleet verse sessie**. Ik heb het geteld: ongeveer **53 wegwerp-sessies per dag**. Die verse bot krijgt precies dit te zien:

> Dit is een opdracht uit de team-wachtrij, van kimi. Jamal typt hier niet mee. Voer hem uit en beschrijf kort wat je gedaan hebt, met bewijs.
>
> --- opdracht ---
> [één zin]

Geen taaknummer. Geen eerdere berichten. Geen "je was hier gisteren al mee bezig". Niets.

Dat is de kern. **De bots vergeten niet. Ze hebben nooit iets gekregen om te onthouden.**

---

2. Wat er al goed gaat

Dit moet ik eerlijk zeggen: je hebt al veel meer staan dan de meeste mensen die hiermee bezig zijn.

- **gBrain is goed gevuld.** 773 pagina's, 1.604 stukjes, allemaal doorzoekbaar, geen dode links. Dat is een echt kennisbrein.
- **Je geheugen staat onder git.** 313 commits. Niets verdwijnt stilletjes. Dit is precies de richting die Letta (het meest doordachte geheugensysteem dat er is) sinds dit jaar zelf ook opgaat: geheugen als markdown-bestanden in een git-repo.
- **De wachtrij zelf is robuust.** 5.314 berichten, realtime plus een vangnet dat elke 2 minuten kijkt, lus-beveiliging, en sinds 31 juli blijft een bericht netjes op "pending" staan in plaats van te verdwijnen.
- **De hooks staan er.** SessionStart laadt bij elke start context in. PreCompact is aangesloten. Dat frame is er dus al.
- **De cockpit werkt** (127.0.0.1:8899) en 8 bots sturen een levensteken.
- **En het belangrijkste:** de tabel `agent_tasks` bestaat al in Supabase, met precies de goede kolommen. `task_description`, `status`, `progress_notes`, `result`, `started_at`. Iemand (jij, eerder) heeft dit al goed bedacht.

Je hoeft dus **niets nieuws te verzinnen**. Je moet afmaken wat er half staat.

3. Waar het gat zit

Ik heb het gemeten, niet geschat. Zes gaten, op volgorde van hoe erg.

a) Je vangnet vangt niets. Dit is de directe oorzaak.

De PreCompact-hook is bedoeld om vlak vóór een compact een verslag weg te schrijven. Hij heeft **111 verslagen geschreven**. In alle 111 staat nul gesprek.

Het nieuwste verslag, volledig:

> Claude compact 2026-07-31
> Checkpoint 00:42 (preCompact)
> (geen nieuwe gespreksinhoud sinds laatste verslag)

Twee regels code zijn de oorzaak, in `/Users/macbookjamal/.claude/scripts/session-compact-report.py`:

- Regel 127 wijst naar `.claude/projects/-Users-imacjamal`. Die map bestaat niet op de MacBook. Het is het oude iMac-pad.
- Regel 153 zoekt het veld `role`. In de sessiebestanden heet dat veld `message.role`. Ik telde in de sessie van 31 juli: **0 regels met `role`, 3.312 met `message.role`**.

En ondertussen meldt de hook op je scherm keurig "automatisch verslag opgeslagen". Een groen vinkje op een leeg bestand. Precies de val waar je eigen regelboek voor waarschuwt.

b) Een taak heeft geen naam.

In `agent_messages` is geen `task_id`, geen `parent_id`, geen verwijzing naar het bericht ervoor. Van de 5.314 rijen heeft 6% iets in `metadata`, en dat is puur Telegram-routering.

Elk bericht staat volledig op zichzelf. Er is geen enkele manier om te vragen: waar hoort dit bij.

c) De opdracht is één zin, zonder verleden.

Zie hierboven. In `queue-worker.js` (regel 313 tot 318) staat bewust "geen `--continue`". Dat is een verdedigbare keuze, maar er is niets voor in de plaats gekomen.

d) Het gedeelde brein wordt niet gebruikt.

Ik heb alle 343 wachtrij-sessies doorgeteld:

TOOL	KEER GEBRUIKT	IN HOEVEEL SESSIES
Bash	1.854	293
gbrain zoeken	12	5
gbrain een feit erin zetten	**1**	**1**

In 343 verse sessies is er **één keer** een feit in gBrain gezet. De regel bestaat. Het gedrag niet.

e) Het geheugen komt niet binnen.

De SessionStart-hook laadt van elke MEMORY.md maximaal 3.000 tekens. Jouw MEMORY.md is 19.919 bytes. Dat is **15%**. En van de 517 losse geheugenbestanden staan er 356 (69%) nergens in de index.

f) De machinekaart klopt niet, en dat verklaart de vraag letterlijk.

De vraag "welke machine is McMini Veranda?" komt hier vandaan:

- De registry kent **geen aliassen**. In de registry heet hij `coco-veranda`, in de heartbeats `veranda`, en jij zegt "McMini Veranda".
- In gBrain staat het IP **192.168.1.43**. Ik heb ingelogd en gemeten: het echte IP is **192.168.1.2**.
- Paco's kopie van de registry is van **18 juli**. Dertien dagen oud.

En Paco is het zwaarst getroffen: 296 van zijn 558 berichten zijn nooit verwerkt. Dat is **53%**.

---

4. De oplossing, in lagen

Belangrijk vooraf: de drie ideeën uit de briefing zijn eigenlijk **twee** ideeën.

"Context-persistentie" en "unieke taak-ID's" zijn hetzelfde ding. Het taak-ID **is** de sleutel waarmee je de opgeslagen context terugvindt. Zonder ID geen persistentie.

Idee 3, "Conductor bewaart de context en stuurt die mee", is het minst robuust in de sociale vorm. Als Conductor bij elke stap de hele geschiedenis meestuurt, groeit elke opdracht mee

en loop je precies tegen dat "lost in the middle"-probleem aan. Beter: **Conductor bewaart, de bot haalt op via het ID.**

Laag 0: de gratis winst (vandaag, samen ongeveer 1 uur)

0.1 De PreCompact-hook repareren. Twee regels. Pad naar `-Users-macbookjamal`, veld naar `message.role`. Daarna één keer testen met een echte compact en het verslag lezen. Dit alleen al zorgt dat er straks überhaupt iets is om terug te lezen.

0.2 Een blok "Compact instructions" in elke CLAUDE.md. Dit is officieel ondersteund en wordt ook bij een automatische compact gebruikt. Voorbeeld:

```
# Compact instructions
Bewaar bij het samenvatten altijd letterlijk: het taak-ID, de
oorspronkelijke opdracht woord voor woord, wat al af is met bewijs,
en wat de volgende stap is.
```

Nu raadt de samenvatter zelf wat belangrijk is. Dit is het enige knopje waarmee je dat stuurt. Kost vijf minuten per bot.

0.3 De kapotte `resize-large-images.sh`-hook weghalen. Het script bestaat niet meer, de verwijzing staat er nog. Kost **420 mislukte hooks per 30 dagen**. Vijftien minuten.

0.4 Het foute IP in gBrain corrigeren. 192.168.1.43 wordt 192.168.1.2.

Laag 1: de taak krijgt een naam (een halve dag)

Twee tabellen in Supabase. Meer niet.

Tabel 1: `agent_taken` (één rij per opdracht, overleeft alles)

VELD	WAT ERIN STAAT
`id`	`HV-20260801-014`, leesbaar, kun je hardop noemen
`opdracht`	de originele tekst, letterlijk, **wordt nooit overschreven**
`opdrachtgever`	jamal, conductor, cron
`toegewezen_aan`	paco
`status`	wachtend, bezig, wacht_op_mens, klaar, mislukt

VELD	WAT ERIN STAAT
`klaar_wanneer`	wat "af" betekent, in gewone taal
`context`	vaste feiten: welke machine, welk pad, welke url
`volgende_stap`	één zin: wat doe je als eerste na een herstart
`pogingen`	teller
`lease_tot`	tot wanneer deze bot hem vasthoudt
`bewijs`	link, pad of testuitkomst

Tabel 2: `agent_taak_log` (append-only, alleen bijschrijven, nooit wijzigen)

Per regel: taak-ID, tijd, welke bot, soort (gestart, stap_klaar, geblokkeerd, vraag, bewijs, klaar, fout) en de tekst.

Dat is het hele systeem. Twee tabellen.

Belangrijk detail: **`opdracht` is bevroren.** Precies zoals je golden-master-regel bij beeld. Verduidelijkt jij iets later, dan wordt dat een nieuwe regel in het logboek, geen wijziging van de originele tekst. Anders raak je alsnog kwijt wat er echt gevraagd was.

Laag 2: de worker plakt de briefing erbovenop (een halve dag)

Nu bouwt `queue-worker.js` de prompt uit die tabellen. Elke verse sessie krijgt dit:

```
=== TAAK HV-20260801-014 (poging 2 van 3) ===
Van: Jamal, 1 aug 10:12

Oorspronkelijke opdracht, letterlijk:
  "..."

Klaar is het pas als: <klaar_wanneer>
Vaste feiten: machine = McMini Veranda = keukenhof-veranda = 192.168.1.2

Wat er al gebeurd is:
  10:15  gestart
  10:41  stap klaar, blog staat op /tmp/blog-014.md
```

```
  11:03  geblokkeerd, wacht op tarief van Jamal

Jouw eerste actie nu: <volgende_stap>

Log elke afgeronde stap met:
  taak.py log HV-20260801-014 stap_klaar "..."
```

Let op de rem: **altijd de originele opdracht plus de laatste 5 tot 10 regels plus één samenvattingsregel van al het oudere.** Nooit het hele logboek. Anders verplaats je het probleem alleen maar.

Dit is exact hoe de OpenAI Agents SDK het doet: voor elke run haalt hij de geschiedenis op en plakt die vooraan.

Laag 3: het claimen en het vanzelf terugkomen (2 uur)

Een taak pakken gaat met één Postgres-query met `for update skip locked`. Dat is letterlijk waar die functie voor bedoeld is, staat zo in de Postgres-documentatie. Twee bots kunnen dan nooit dezelfde taak pakken.

En elke claim krijgt een **houdbaarheidsdatum van 20 minuten** (`lease_tot`). Werkt de bot door, dan verlengt hij hem bij elke log-regel. Crasht Paco, dan komt de taak na 20 minuten **vanzelf** terug op "wachtend". Geen mens nodig.

Dit is het visibility-timeout-patroon van Amazon SQS. Bewezen, saai, werkt.

Dit lost meteen je 105 stille "pendings" op, waarvan de oudste van 5 juli is.

En het lost dit op: als je wilt weten of Paco vastloopt, **vraag je hem niets meer**. Een verse bot zegt altijd "ik heb geen actieve taak", want hij ziet zijn eigen parallelle werk niet. Je kijkt gewoon in de tabel:

```
select id, titel, status, lease_tot
  from agent_taken
 where toegewezen_aan = 'paco' and status = 'bezig';
```

Meten aan de buitenkant. Precies wat je eigen regelboek al voorschrijft.

Laag 4: de hooks maken het rond (2 uur)

- **SessionStart met matcher `compact`.** Na een compact start Claude Code opnieuw op met bron "compact". Op dat moment duw je via `additionalContext` het taak-ID en de

originele opdracht terug naar binnen. Dit is precies het gat waar Paco in valt.

- **PostCompact-hook.** Die bestaat, wordt bij jou nergens gebruikt, en krijgt het veld `compact_summary`, de daadwerkelijke samenvatting. Die schrijf je weg bij het taak-ID. Dan kan de bot ná een compact teruglezen wat er is gebeurd.
- **SessionStart-afkapping verhogen** van 3.000 tekens naar het hele bestand, of MEMORY.md slanker maken. Nu komt 15% binnen.

Laag 5: de discipline (doorlopend)

- **Registry uitbreiden met aliassen**, hostnames, lokale IP's en "welke bot draait hier". Daarna automatisch naar alle bots syncen, want Paco's kopie is 13 dagen oud.
- **Één statuslijst afdwingen** met een check-constraint. Nu zijn er al 9 verschillende statuswaarden in gebruik, waaronder drie opruim-labels. Zonder rem verzint elke bot zijn eigen woord.
- **Werkverdeling helder houden:** gBrain = feiten die blijven gelden (wie is Asuman, welke machine is Coco). `agent_taken` = wat er nu loopt. Een afgeronde taak schrijft één samenvatting naar gBrain en verder niets.

---

5. Wat ik zou bouwen, deze week

Dag 1, ochtend: de gratis winst (1 uur)

STAP	WERK	LEVERT OP
PreCompact-hook repareren, 2 regels	30 min	het vangnet vangt eindelijk iets
Compact-instructies in elke CLAUDE.md	15 min	de samenvatting bewaart voortaan de opdracht
Kapotte resize-hook weghalen	15 min	420 minder fouten per maand
Fout IP in gBrain corrigeren	5 min	minder verwarring over de veranda

Jouw goedkeuring nodig: nee, dit zijn reparaties van dingen die al kapot zijn.

Dag 1, middag: dezelfde check op alle machines (2 uur)

Ik heb alleen de MacBook kunnen meten. Of `session-compact-report.py` bij Paco, Sjakie, Coco, Kimi, Lenovo, Julio en Appel hetzelfde iMac-pad heeft staan, **weet ik niet**. Dat moet per machine gecontroleerd worden voordat iemand zegt dat het overal stuk is.

Jouw goedkeuring nodig: nee.

Dag 2: de twee tabellen plus `taak.py` (1 dag)

De tabellen aanmaken, plus één klein script met vier commando's: `claim`, `log`, `klaar`, `blokkeer`. Ongeveer 200 regels code.

Ik zou de bestaande dode tabel `agent_tasks` **niet hergebruiken** maar netjes archiveren en één nieuwe maken met de goede kolommen en een constraint. Anders heb je straks twee halve systemen.

Jouw goedkeuring nodig: ja. Dit is een nieuwe tabel in Supabase. Zeg alleen ja of nee.

Dag 3: de worker ombouwen (1 dag)

`queue-worker.js` bouwt de briefing uit de tabel. Weigert een opdracht zonder taak-ID en maakt er desnoods zelf een aan.

Dit is de harde regel die maakt of het werkt: **er is één pad.** Niet twee systemen naast elkaar. Precies dat is de reden dat `agent_tasks` sinds maart dood ligt en `task-claim.py` sinds 9 juni niet meer draait.

Jouw goedkeuring nodig: ja. Dit raakt de wachtrij die nu werkt. Ik zou het eerst één dag alleen bij Paco aanzetten, en de rest ongemoeid laten.

Dag 4: hooks en registry (halve dag)

SessionStart met compact-matcher, PostCompact-logging, registry met aliassen plus sync naar alle bots.

Jouw goedkeuring nodig: nee.

Dag 5: meten of het werkt

Niet "klaar" zeggen omdat er geen foutmelding kwam. Wat ik meet:

- Paco's onverwerkte berichten: nu 296 van 558. Doel onder 10%.
- Aantal keer dat Conductor met de hand context nastuurt: nu ongeveer 3 keer per dag.
- Aantal taken dat op "bezig" blijft hangen met een verlopen lease: doel nul.
- Een echte compact uitlokken en het verslag lezen. Staat er inhoud in, ja of nee.

Totaal: ongeveer 3 dagen werk. Twee goedkeuringen van jou nodig.

---

6. Wat ik niet zou doen

Geen Temporal, Restate of AWS Step Functions. Dat zijn de zware jongens voor "een taak overleeft een crash". Ze werken prima, maar ze vragen een eigen server, workers, versiebeheer van workflowcode en een hoop discipline. Voor 8 bots is dat niet te verdedigen. Twee tabellen geven je 90% van het voordeel.

Geen full replay. Temporal herstelt door de hele code opnieuw af te spelen. Dat werkt alleen als de code altijd exact hetzelfde doet. Een taalmodel doet dat per definitie **niet**. Wie dit toch probeert, bouwt iets dat structureel breekt. Herspeel alleen **uitkomsten** ("blog staat op pad X"), nooit de redenering.

Geen Mem0, Letta, Zep of Cognee erbij. Allemaal goed. Maar dat zijn **kennisgeheugens**, en dat heb je al: gBrain, met 773 pagina's. Een tweede kennisbrein naast het eerste geeft je twee bronnen die uit elkaar gaan lopen. Je probleem is niet kennis. Je probleem is taak-geheugen.

Geen LangGraph. Het onderliggende idee (één ID als sleutel naar opgeslagen state) neem ik wel over, want dat is de standaard. Maar het framework zelf betekent je hele wachtrij herbouwen. Niet nodig.

Niet de hele geschiedenis meesturen. Dat voelt veilig en is het niet. Hoe langer de tekst, hoe slechter het model het midden leest. De briefing moet kort blijven.

Niet vertrouwen op benchmarkcijfers van leveranciers. Ik ben ze tegengekomen en ze spreken elkaar tegen. Zep heeft de cijfers van Mem0 uitgebreid onderuitgehaald, en bij die kritiek bleek dat een domme "gooi alles in de context"-aanpak beter scoorde dan beide. Kies dus niet op cijfers.

En het belangrijkste dat ik niet zou doen: nog een systeem bouwen dat niemand afdwingt. Je hebt nu drie dode dingen liggen: `agent_tasks` (leeg sinds maart), `task-claim.py` (dood sinds 9 juni), en `~/.context/todos.md` waar je CLAUDE.md naar verwijst maar dat niet bestaat. Als de worker het niet afdwingt, ligt tabel nummer vier over vier maanden ook stil.

---

7. De andere adviezen uit de briefingen, met status

Ik heb alle briefingen van 24 tot en met 31 juli gelezen en per advies zelf gemeten. In 8 briefingen staan ongeveer **40 losse adviezen**. Daarvan zijn er **3 aantoonbaar uitgevoerd**.

Wel gedaan

ADVIES	BEWIJS
Windows python-fout bij Paco fixen	`python3 --version` geeft nu 3.12.10
Foto-recept-hook uitrollen	Paco 30-7, Coco 31-7, staat in de hooks
SSH naar Paco zodat je zijn werk kunt controleren	werkt, ik heb er zelf bestanden gelezen
Coco op de veranda draaien	draaide al, het advies zelf klopte niet
Udio-brug voor muziek	werkt, 210 credits, account JamalDani
Centraal dashboard bot-status	cockpit geeft HTTP 200

Half gedaan

ADVIES	WAT ER MIST
"Één keer bevestigen"-regel	Regel bestaat sinds 31-7. Maar het queue-verkeer ging van 46 per dag naar **309 per dag**, en Paco stuurde op 31-7 alleen al 140 berichten. Een regel zonder rem werkt hier niet
Machine-registry	Bestaat, maar heeft alleen host en gebruiker. Geen aliassen, geen lokale IP's, geen "welke bot draait hier". 6 machines erin terwijl Tailscale er 14 toont

ADVIES	WAT ER MIST
Mijlpalen bij alle bots	Conductor 136, Paco 13, Sjakie 7, Coco 1. Kimi, Julio, Lenovo en Appel: **nul**. Plus Conductor staat er 3 keer in met verschillende schrijfwijzen
Conductor als contextbeheerder	Gebeurt met de hand, ongeveer 3 keer per dag. Bericht 4660 zegt letterlijk "de context die je mist, je vroeg er twee keer om"

Niet gedaan

ADVIES	WERK	OPMERKING
`shared_memory_client.py` naar Paco	1 uur	staat op de MacBook, niet op Paco
Meldingsafspraak 2x per dag bevestigen bij alle bots	1 uur	geen teamregel bevat dit
Taakverdeling vastleggen (Paco = audio, Sjakie = beeld)	2 uur	staat alleen los in één queue-bericht
Registry naar alle bots syncen	1 uur	Paco's kopie is van 18 juli
Git bij Paco	2 uur	nul git-mappen in zijn hele thuismap, terwijl dit een teamregel is
Wachtwoordkluis voor bots	4 uur	Lenovo's monitor ligt al 2 dagen stil op één ontbrekend bestand
Beslissings-termijn afspreken	2 uur	Sjakie wacht **7 dagen** op jouw oordeel over scherm_v9.mp4
Cloud-opslag zodat je niet zoekt waar bestanden staan	1 dag	geen spoor

De adviezen over de salon die elke keer terugkomen

Deze staan in bijna elke briefing en zijn nog nooit opgepakt:

- **Voice-AI of AI-receptioniste voor de salon: 6 keer herhaald**
- **AI beeld en video voor social: 6 keer**

- **Virtuele kapsel- en kleurspecialist upgraden: 4 keer**
- Keratine- en extension-nieuws: 3 keer
- Smart home Spanje: 2 keer

Eén daarvan heeft een klantveiligheids-kant en verdient wel aandacht: het bericht over **chemicaliën in hairextensions** (30-7). Let op, ik heb de bronnen daarvan **niet nagelezen**. Behandel het als onbevestigd tot iemand het echt controleert.

Het meta-probleem

Op 30 juli vroeg je aan Jarvis: "er komen elke briefing adviezen langs, maar gebeurt daar ook echt iets mee?"

Er kwam een actieplan met 4 punten. Alle 4 stonden op "openstaand". Twee dagen later heb ik ze nagemeten: **3 van de 4 staan er nog steeds zo bij.**

Het actieplan is dus zelf het bewijs van het probleem.

Mijn advies daarop is simpel: **stop met adviezen in briefingen zetten zonder taak-ID.** Elk advies dat werk vraagt, wordt een rij in `agent_taken` met een eigenaar en een datum. Anders schrijven we volgende week hetzelfde op.

Wat ik niet heb kunnen controleren

Eerlijk zijn hierover is belangrijker dan een compleet verhaal.

- Ik heb **alleen de MacBook gemeten**. De hooks, scripts en sessiebestanden van Paco, Sjakie, Coco, Kimi, Lenovo, Julio en Appel heb ik niet gezien. De briefing gaat over Paco, mijn metingen gaan over Conductor. Of de PreCompact-hook daar ook stuk is, weet ik niet.
- Ik heb **niet gemeten hoe vaak Paco's sessies echt compacten**. Dat kan alleen op Paco zelf.
- Er bestaat **geen enkel onderzoekscijfer** over hoeveel beter een vloot van 8 bots wordt van taak-persistentie. Alle cijfers die ik vond gaan over één agent of over chatgeheugen. Wie je een percentage belooft, verzint het.
- De ElevenLabs-sleutel veilig opslaan: de commando's daarvoor werden geblokkeerd. Weet ik niet.

Nog iets dat ik zag en moet melden

De tabel `bot_heartbeats` in Supabase heeft **RLS uitgeschakeld**. Iedereen met de publieke sleutel kan die lezen en wijzigen.

Aanzetten is één regel SQL, maar zonder regels erbij blokkeert dat meteen **alle** toegang, ook die van je eigen bots. Dat moet jij bewust beslissen. Zeg het maar, dan zet ik het met de juiste regels erbij.

Bestanden die er bij een fix toe doen

- `/Users/macbookjamal/.claude/scripts/session-compact-report.py` regel 127 en 153, de kapotte hersteller
- `/Users/macbookjamal/conductor-queue/queue-worker.js` regel 313 tot 318, de context-arme prompt
- `/Users/macbookjamal/.claude/scripts/session-start-context.py` regel 84, de afkapping op 3.000 tekens
- `/Users/macbookjamal/.claude/scripts/sessie-omvang.py` regel 88 tot 90, ziet automatische compacts niet en meet daardoor verkeerd
- `/Users/macbookjamal/.claude/scripts/task-claim.py`, dood sinds 9 juni
- `/Users/macbookjamal/.claude/machine-registry.json`, 6 van de 14 machines, laatst bijgewerkt 28 juli

In één alinea

Je bots vergeten niets, ze krijgen nooit iets mee. Elke opdracht start een verse sessie met één losse zin en geen verleden. Je vangnet voor compacts is stuk door twee regels code en heeft 111 lege verslagen geschreven. De oplossing is niet een nieuw systeem, maar afmaken wat er half staat: repareer die twee regels vandaag, geef elke taak een nummer, en laat de wachtrij-worker bij elke start de originele opdracht plus de laatste stappen erbovenop plakken. Drie dagen werk, twee keer jouw ja.