

Hermes Playbook for Scot

The Hermes Playbook — getting real mileage out of the harness

Prepared for Scot (Syslog Solution LLC). Version: Hermes Agent v0.21.1. Every CLI command below was verified live against that version on a reference install (Syslog kagentz) on 2026-09-11; anything only confirmed against the official docs is tagged DOC-ONLY.

You are already running Hermes next to Claude Code, on your own OpenRouter account with fast models (qwen3.8-flash, deepseek-4-flash). The question you asked: why does Hermes feel like it has less context, and what do I do about it?

1. TL;DR

- The context gap is not a bug. Claude Code reads the repo it sits in on every launch; a fresh Hermes install starts nearly empty by design. It gets its context from files you seed and from memory it builds over time.
- One command closes most of the gap on day one: `hermes import-agent claude-code` carries your CLAUDE.md instructions, MCP servers, skills, and memories into Hermes (preview first with `--dry-run`).
- Teach Hermes once, and it remembers: "save this as a skill" after any workflow you repeat. Skills auto-load when a matching task comes up — that is the learning loop.
- Keep per-project context in an `AGENTS.md` in the repo root (git-tracked, shared with your team) and personal preferences in your persona file and persistent memory.
- Hermes and Claude Code are not rivals: let Hermes be the always-on orchestrator (research, briefs, scheduling, messaging) and hand heavy coding to Claude Code, which Hermes can drive directly.

2. Why Hermes feels like it has less context (and why that is fixable)

Honest comparison, no spin:

	Claude Code	Hermes (fresh install)
Where context comes from	The repo: <code>CLAUDE.md</code> auto-loaded every launch; <code>.claude/</code> folders with subagents, slash commands, hooks, skills	Config files: <code>AGENTS.md</code> in the working directory + <code>SOUL.md</code> persona + persistent memory from the Hermes home
What it remembers between sessions	<code>~/.claude/projects/<project>/memory/</code> (25 KB cap)	First-class persistent memory, always injected — <code>MEMORY.md</code> / <code>USER.md</code> plus optional external providers
	You write the skill/command files	

	Claude Code	Hermes (fresh install)
How it learns your workflows		It can write its own skills after learning a workflow, and a curator maintains them
Out-of-the-box feel	Context-rich if you have invested in your CLAUDE.md	Quiet until you seed it

That last line is the whole story. Claude Code's context is the sum of everything you built in `CLAUDE.md` and `.claude/` over months. A fresh Hermes has none of that yet — not because the harness is weaker, but because it stores context in different places and expects you to seed it (or let it build up).

The gap is fixable in two moves:

1. Import what you already have. `hermes import-agent claude-code` maps `CLAUDE.md/AGENTS.md` instructions, permission allowlists, MCP servers, skills, and memories into Hermes equivalents. Preview with `--dry-run`; it never imports API keys; conflicts are skipped by default (`--overwrite` to change).
2. Let the learning loop run. Every time you correct Hermes or finish a workflow you will repeat, tell it to remember. Within a few weeks it will have its own `CLAUDE.md` equivalent — built, not typed.

What the comparison table in our research covers, gap by gap: project instructions, instruction splits, slash commands, subagents, skills, project memory, tool permissions, MCP, session resume, cost/context visibility, headless mode, prior-setup import, hooks, and scheduled work. Each has a Hermes equivalent, and every one is documented in section 9.

3. The context stack

This is the order in which Hermes builds its context, and what you do at each layer.

Layer 1 — Persona (`SOUL.md`). Set up once. Your Hermes' standing identity and voice: "you are my analyst," the tone, the standing rules. Auto-injected into every session. Lives at `~/.hermes/SOUL.md` (per profile: `~/.hermes/profiles/<name>/SOUL.md`). Docs: <https://hermes-agent.nousresearch.com/docs/user-guide/configuration>

Layer 2 — Persistent memory. Set up once, then feed it constantly. `MEMORY.md` / `USER.md` are always active and injected every session — this is the single biggest cure for "it forgets my project." After any correction or preference ("use this source list," "briefs go in this format"), tell Hermes to remember it. Manage with `hermes memory setup|status|off|reset` (VERIFIED-LIVE). Optional external providers exist (Honcho, Mem0, and others). Docs: <https://hermes-agent.nousresearch.com/docs/user-guide/features/memory>

Layer 3 — Skills. Set up once; grows forever. Markdown procedure files that auto-load when a task matches the skill. The differentiator: after completing a workflow, ask Hermes to "save this as a skill" — it authors the skill itself, and a background curator tracks usage, archives stale ones, and keeps backups. CLI: `hermes skills list|search|install|browse|config|check|update` (VERIFIED-LIVE); in-session: `/skill <name>`, `/reload-skills` (DOC-ONLY). Docs: <https://hermes-agent.nousresearch.com/docs/reference/skills-catalog> and <https://hermes-agent.nousresearch.com/docs/user-guide/features/curator>

Layer 4 — Projects. Per workstream. `AGENTS.md` in each repo root (git-tracked, team-shared) carries project rules; Desktop Projects (`hermes project create <name>` then `add-folder`) group multi-repo work under one named workspace. Both VERIFIED-LIVE.

Layer 5 — Retrieval (session store). Automatic. All conversations land in a searchable store; Hermes can search past sessions when you ask "what did we decide last week." CLI: `hermes sessions list|browse|rename|pin|export|prune|stats` (VERIFIED-LIVE).

Layer 6 — MCP (external tools). Per integration. Plug GitHub, databases, workflow engines into the agent. `hermes mcp add|list|test|configure|picker|catalog|install` (VERIFIED-LIVE). Docs: <https://hermes-agent.nousresearch.com/docs/user-guide/features/mcp>

Quick summary:

Layer	Set up	Feed
SOUL.md persona	once	rarely
Persistent memory	once	every correction/preference
Skills	once	"save this as a skill" after repeated workflows
AGENTS.md / Projects	once per repo/workstream	as projects evolve
Session retrieval	automatic	ask
MCP	once per integration	when new tools appear

4. Top moves

The highest-leverage moves for your kind of work — research, evidence-graded analysis, weekly briefs — and for running alongside Claude Code. Every command verified on v0.21.1.

1. Import your Claude Code setup

```
hermes import-agent claude-code --dry-run # preview
hermes import-agent claude-code          # migrate CLAUDE.md, MCP, skills, memories
```

What it does: one-command migration of the instructions and servers that made Claude Code feel context-rich, translated into Hermes equivalents. Never imports API keys. Why it matters: this is the direct answer to "Hermes has no context." After this, Hermes knows your projects on day one.

2. Bring over the conversation history

```
hermes sessions import
```

What it does: imports Claude Code or Codex CLI conversations into the Hermes session store. Why it matters: mid-project, the new agent picks up exactly where the old one left off. `hermes --resume <id>` and `hermes sessions browse` then treat the history as native.

3. Trust your repos so project-local skills load

```
hermes skills trust
```

What it does: trusts a repo so its project-local skills (`.hermes/skills`) load — the Hermes analog of `.claude/skills/` . Why it matters: your evidence-grading rules can live in the repo with the project, versioned with git, and load automatically.

4. Per-directory session continuity

```
hermes --in DIR --resume latest
```

What it does: resumes the latest session for a given directory (also: `hermes -c [NAME]` , `hermes --resume <id|latest>`). Why it matters: every project folder gets its own continuous thread. Research on one portfolio never mixes with another.

5. Preload skills for a specific job

```
hermes -s skill1,skill2
```

What it does: preloads specific skills for the session. Why it matters: for a weekly brief or an evidence register, pin the exact skills that encode your grading criteria instead of hoping they auto-match.

6. Save any repeated workflow as a skill

In-session: "save this as a skill." (CLI: `hermes skills list|search|install|browse` .)

What it does: Hermes authors a skill file from the workflow you just ran. Why it matters: the learning loop is the whole point. Do the evidence-grading pass twice, save it, and every future run starts with the procedure loaded.

7. Fan out research with subagents

In-session: "delegate this to subagents." (agent-side tool `delegate_task` , no CLI.)

What it does: parallel subagents with isolated contexts — each gets its own conversation and terminal, only the final summary comes back. Why it matters: research fan-out without flooding your main context. Ten sources, ten subagents, one synthesis.

8. Make the weekly brief a cron job

```
hermes cron create
```

What it does: durable scheduler — duration or cron syntax, per-job model overrides, output chaining, delivery to messaging platforms. Manage with `hermes cron list|create|edit|pause|resume|run|remove|doctor` . Why it matters: a weekly brief is exactly a cron job. It runs even when you are not at the desktop, with your skills preloaded and its output delivered to you.

9. Set a standing goal for grind work

In-session: `/goal [text|status|pause|resume|clear]` (DOC-ONLY; CLI subcommands verified).

What it does: a standing objective the agent keeps working toward across turns until achieved. Why it matters: "keep researching until you have 5 verified sources" — the agent loops itself instead of waiting for you to say "go on."

10. Run Hermes as an MCP server for Claude Code

```
hermes mcp serve
```

What it does: exposes Hermes (persistent memory, skills, cron, sessions) to other agents as an MCP tool provider. Claude Code supports MCP clients, so it can consume Hermes. Why it matters: the reverse bridge. Claude Code gets the surfaces it lacks, and both tools share your knowledge base.

11. Pick the right model per task, with a safety net

```
hermes fallback list|add|remove
hermes -m MODEL --provider PROVIDER --reasoning high
```

What it does: explicit fallback chains (a failed call rolls to a second model instead of erroring) and per-run model/provider/reasoning overrides. Why it matters: on OpenRouter with fast models, use `--reasoning high` for the hard analytical passes and let fallback chains keep the cheap models from stalling your brief.

12. Diagnose why responses feel thin

```
hermes prompt-size
```

What it does: byte breakdown of the system prompt + tool schemas. Why it matters: when output quality drops, it is usually context bloat, not model quality. This tells you what is eating the window.

5. Working alongside Claude Code

You run both. The proven patterns, in order of value.

First: import. `hermes import-agent claude-code` then `hermes sessions import`. After this, the "two tools that don't know each other" problem is gone — Hermes knows your projects and your history.

Hermes as orchestrator, Claude Code as worker. The installed Hermes skill for exactly this is `autonomous-ai-agents/delegate-coding-agent`. Two modes:

- Print mode (preferred): `claude -p '<task>' --allowedTools 'Read,Edit' --max-turns 10` — one-shot, no dialogs, structured JSON output with `session_id`, `num_turns`, `total_cost_usd`. In Hermes, just say: "delegate this coding task to Claude Code in print mode."
- Interactive PTY via tmux: Hermes starts a tmux session, sends prompts with `send-keys`, monitors with `capture-pane`. For iterative refactor → review → fix cycles.

There is also a cross-agent review loop: `git diff main...feature | claude -p 'Review this diff for bugs and security issues.' --max-turns 1` — Hermes runs it, reads the findings, and fixes them itself. Claude Code

becomes a reviewer Hermes coordinates. The skill's safety rails: explicit workdir, clean git status before launch, narrow task prompts, git diff review, targeted tests before committing.

Parallel workstreams, neutral merge reconciliation. When both agents edit the same repo and collide, do not let either resolve the conflict — both are biased toward their own side. Spawn a neutral third agent with the `merge-reconciler` skill: it classifies every conflicted hunk, resolves under an impartiality contract, verifies with build/tests, and hands back a summary naming every decision. Kanban shape: a reconciliation card assigned to a third profile, with both workers' cards as parents.

Hermes as MCP server (the reverse direction). `hermes mcp serve` (pattern in section 4, move 10). The only bridge direction Claude Code cannot offer.

Desktop GUI goes to Hermes. Claude Code has no desktop automation. `hermes computer-use install` (cua-driver; health check `hermes computer-use doctor`) drives native desktop apps background-first — never steals focus. If a task needs Excel or a native app, that part routes to Hermes while the code routes to Claude Code.

Division of labor in one line: Hermes is the always-on layer — research, briefs, scheduling, messaging, memory, and the orchestration desk. Claude Code is the deep coding worker. Hand coding-heavy tasks over; hand continuity, recall, and scheduled work to Hermes.

6. Video watch list

Every link verified via the YouTube oEmbed endpoint on 2026-09-11 (status PASS, title/author matched). All content is third-party ecosystem material — no official Nous Research tutorial video exists (see section 8).

#	Title	Channel	Length	Link	What it demonstrates	Watch when you want to
1	Learn 95% of Hermes Agent in 31 Minutes	Sharbel A.	31:28	https://www.youtube.com/watch?v=Ta2wg6xPaY4	End-to-end fundamentals: install, sessions, skills, memory, the learning loop	the fastest real overview of the whole harness before touching config
2	Hermes Agent Fundamentals In 29 Minutes	Tina Huang	29:40	https://www.youtube.com/watch?v=5_N84t1rUU0	Why Hermes' memory/skills loop differs from one-shot coding agents	understand why Hermes feels different from Claude Code
3	Every Level of Hermes Agent Explained	Jack Roberts	25:35	https://www.youtube.com/watch?v=6GtF_uHbGhw	Beginner to advanced ladder: memory, skills, automation, multi-agent	a map of what to learn next after the basics
4	Hermes Agent Full Tutorial INSTALLATION + USECASES	CodeHead	7:47	https://www.youtube.com/watch?v=8Gjy0Qy19so	Install through real use cases, compact	a quick install-to-value demo to share with a colleague
5		CodeHead	4:53	https://www.youtube.com/		

#	Title	Channel	Length	Link	What it demonstrates	Watch when you want to
	Hermes Agent Explained In 5 Minutes			watch?v=9GpWELm3_XI	Conceptual pitch of the agent and its learning loop	the elevator pitch before committing 30 minutes
6	100 Days With Hermes Agent in 21 Minutes	Sharbel A.	21:19	https://www.youtube.com/watch?v=sCa3BtpkziQ	What memory/skills accumulation looks like after months of daily use	see the payoff of the learning loop over time
7	Hermes Agent - Crash Course for Beginners (AI Agent)	Adrian Twarog	22:19	https://www.youtube.com/watch?v=4sAmpcSOVEw	Beginner crash course from a well-known dev channel	a second independent explanation of the basics
8	Hermes Agent: The Ultimate Beginner's Guide	Metics Media	37:08	https://www.youtube.com/watch?v=CwPUOVUdApE	Long-form beginner guide incl. setup and everyday workflows	the most thorough single walkthrough in one sitting
9	Hermes Agent Just Killed OpenClaw (Full Tutorial)	Leon van Zyl	19:59	https://www.youtube.com/watch?v=jmtpYUOr7_U	Feature-by-feature tutorial (MCP config, memory, agents)	a practitioner's feature-by-feature tutorial
10	Hermes Agent vs OpenClaw	Sharbel A.	15:28	https://www.youtube.com/watch?v=zwqhemjHq3E	Head-to-head comparison of the two agent harnesses	the tradeoffs between Hermes and its main alternative
11	Better than OpenClaw? Testing Hermes Agent w/ Qwen 3 model	Tonbi's AI Garage	15:08	https://www.youtube.com/watch?v=8tpuky8HpXw	Hermes driven by an OpenRouter-served open model	how small open models behave inside Hermes
12	Use This To Make The Hermes Agent Basically Free	AI LABS	13:08	https://www.youtube.com/watch?v=5d02TYo0zfE	Running Hermes on cheap/free model backends	cut inference costs on an OpenRouter account
13	Hermes Agent The 24/7 Self-Evolving AI Agent!	WorldofAI	9:15	https://www.youtube.com/watch?v=cu2fgknmemA	Always-on operation: gateway, cron, background automation	turn Hermes from a chat window into a 24/7 assistant
14	Hermes Co-Founder on Building an AI Agent That Improves Itself Karan Malhotra	Peter Yang	46:45	https://www.youtube.com/watch?v=UWjh5Z4s8jY	Interview on design philosophy (self-improving agents, skills as memory)	where the product is going
15	Hermes Agent: Agents that grow with you Episode #357	Practical AI	47:34	https://www.youtube.com/watch?v=UTZhvPXnmwA	Podcast-depth technical discussion of the agent architecture	the engineering story behind the learning loop
16		Alex Finn	13:55			

#	Title	Channel	Length	Link	What it demonstrates	Watch when you want to
	Did Hermes Agent just kill OpenClaw? (full guide)			https://www.youtube.com/watch?v=tP6yf220Jdl	Guide-style comparison/switch content	a switcher's guide perspective
17	Hermes Agent: Why Everyone's Ditching OpenClaw in 2026	Luke Alexander AI	18:03	https://www.youtube.com/watch?v=1UgXUjT-Qtl	Comparison content	more comparison context

Suggested order: 1 or 5 first (whichever mood you are in), then 2, then 6 once you have a few weeks of use under your belt.

7. Your first 7 days

One action per day, each finishable in 15 minutes.

Day 1 — Import. `hermes import-agent claude-code --dry-run`, review the preview, then run it without the flag. Your CLAUDE.md context now lives in Hermes.

Day 2 — Write your SOUL.md. Open `~/hermes/SOUL.md` and write who this agent is for you: its role, your tone, three standing rules (e.g., how to grade evidence, where briefs go, how to flag uncertainty). Ten lines is plenty.

Day 3 — Per-directory sessions. Pick your most active project folder. Work one task there via `hermes --in DIR --resume latest`. Notice the thread is separate from everything else.

Day 4 — First skill. Finish a small repeated workflow (a source-check pass, a brief section). At the end, say "save this as a skill." Next day, watch it load by itself.

Day 5 — One cron job. `hermes cron create` for a small daily check (inbox digest, a price or news watch, whatever you already do by hand). Deliver it somewhere you actually look.

Day 6 — Hand a task to Claude Code. In Hermes: "delegate this coding task to Claude Code in print mode." Read the JSON result. This is the bridge working.

Day 7 — Recall test. Ask Hermes "what did we decide last week about [your project]?" If it can answer from the session store, the stack is working. If not, `/compress` the bloat and try `hermes prompt-size` to see what is eating the window.

8. What NOT to expect

- A bigger context window than you have. Model choice does not change the window size. Fast models on OpenRouter (qwen3.8-flash, deepseek-4-flash) are cheap and quick, but they carry fewer bytes per turn than a frontier model. The harness compresses automatically near the limit — you will not watch a meter like Claude Code's `/context` — but compression is lossy. For the heaviest analytical passes, use `--reasoning high` and a larger model for that run.

- Model choice as a silver bullet. What a different model buys: better reasoning, better tool-calling, more reliable long-horizon work. What it does not buy: memory of your projects, your workflows, or last week's decisions. That lives in your context stack, not the model.
- Desktop = everything. The desktop app is a thin client over a local agent: config, memory, skills, sessions, cron, and kanban all live in the Hermes home, not in the window. Close the window and the work keeps living; that is a feature, not a bug.
- Cron limits. Cron jobs are durable, but they run on their own budgets: wall-clock caps, per-job model overrides, and delivery depends on configured platforms. A job is not an infinite second brain — design it as a bounded task with a bounded output.
- It will still need to be told things twice. If you did not save it as memory or a skill, the next session does not know. The learning loop only works if you trigger it. "Remember this" and "save this as a skill" are deliberate moves, not magic.
- Official tutorial videos. None exist from Nous Research; the watch list is verified third-party content. The docs (hermes-agent.nousresearch.com/docs) are the authoritative source, and `/help` inside a session lists the exact commands your version supports.
- Slash commands behave like the CLI does. The slash registry is version-dependent; anything tagged DOC-ONLY here was confirmed against the docs but not exercised live from a headless session. `/help` in your own session is the final word.

9. Appendix: command reference

Tags: VERIFIED-LIVE = confirmed against `hermes --help` / `hermes <cmd> --help` on v0.21.1 (2026.9.7), reference install (Syslog kagentz), 2026-09-11. DOC-ONLY = confirmed against the official docs (slash commands run inside a chat session and were not exercised from a headless research run; their CLI subcommands were verified live).

Setup & health

Command	What it does	Tag
<code>hermes setup</code>	Interactive setup wizard	VERIFIED-LIVE
<code>hermes doctor [--fix] [--live]</code>	Diagnose config/deps; <code>--fix</code> auto-repairs	VERIFIED-LIVE
<code>hermes status [--all] [--deep]</code>	Component status	VERIFIED-LIVE
<code>hermes config show/edit/get/set/unset/path/env-path/check/migrate</code>	View/edit config	VERIFIED-LIVE
<code>hermes update</code>	Update Hermes to latest	VERIFIED-LIVE

The Claude Code bridge (highest value for you)

Command	What it does	Tag
		VERIFIED-LIVE

Command	What it does	Tag
<code>hermes import-agent claude-code [--dry-run] [--overwrite] [--yes]</code>	One-command import of a Claude Code setup: maps CLAUDE.md/ AGENTS.md instructions, permission allowlists, MCP servers, skills, memories into Hermes equivalents. Never imports API keys.	
<code>hermes import-agent codex</code>	Same for Codex CLI setups	VERIFIED-LIVE
<code>hermes sessions import</code>	Import a Claude Code or Codex CLI session/conversation into Hermes	VERIFIED-LIVE
<code>hermes skills trust</code>	Trust a repo so its project-local skills (<code>.hermes/skills</code>) load — the Hermes analog of <code>.claude/skills/</code>	VERIFIED-LIVE

Daily driving

Command	What it does	Tag
<code>hermes / hermes chat</code>	Interactive session	VERIFIED-LIVE
<code>hermes -c [NAME] / hermes --resume <id latest></code>	Resume by name or ID	VERIFIED-LIVE
<code>hermes --in DIR --resume latest</code>	Resume the latest session for a directory	VERIFIED-LIVE
<code>hermes -z "PROMPT"</code>	One-shot: prints ONLY the final answer (scripting/CI); tools, memory, and AGENTS.md still load	VERIFIED-LIVE
<code>hermes chat -q "PROMPT"</code>	Single-query mode	VERIFIED-LIVE
<code>hermes -m MODEL --provider PROVIDER --reasoning LEVEL</code>	Per-run model/provider/reasoning overrides (<code>none...ultra</code>)	VERIFIED-LIVE
<code>hermes -s SKILL1,SKILL2</code>	Preload specific skills for the session	VERIFIED-LIVE
<code>hermes -t TOOLSETS</code>	Restrict toolsets for this run	VERIFIED-LIVE
<code>hermes -w</code>	Isolated git worktree session (parallel agents on one repo)	VERIFIED-LIVE
<code>hermes chat --checkpoints</code>	Enable filesystem checkpoints (<code>/rollback</code> to restore)	VERIFIED-LIVE
<code>hermes chat --max-turns N / --run-budget SECONDS</code>	Cap loop iterations / wall-clock budget	VERIFIED-LIVE
<code>hermes -yolo</code>	Bypass command approval prompts (use with care)	VERIFIED-LIVE
<code>hermes pause / hermes resume</code>	Emergency stop / lift (pauses cron, kanban dispatch, gateway turns)	VERIFIED-LIVE

Context & memory management

Command	What it does	Tag
<code>hermes memory setup/status/off/reset</code>	External memory provider management (built-in MEMORY.md/USER.md always active)	VERIFIED-LIVE
<code>hermes sessions list/browse/rename/pin/export/prune/stats</code>	Session store management	VERIFIED-LIVE
<code>hermes skills list/search/install/inspect/browse/config/check/update</code>	Skill management	VERIFIED-LIVE
<code>hermes skills trust/untrust</code>	Repo-local skill trust	VERIFIED-LIVE
<code>hermes curator status/run/pause/pin/...</code>	Background skill maintenance (auto-archive, backups)	VERIFIED-LIVE
<code>hermes prompt-size</code>	Byte breakdown of system prompt + tool schemas (context-bloat diagnosis)	VERIFIED-LIVE
<code>hermes insights [--days N]</code>	Usage analytics	VERIFIED-LIVE

Tools, MCP, integrations

Command	What it does	Tag
<code>hermes tools (interactive) / list/enable/disable</code>	Per-platform toolset toggles; MCP tools as <code>server:tool</code>	VERIFIED-LIVE
<code>hermes mcp add/remove/list/test/configure/picker/catalog/install</code>	MCP server management (incl. one-click catalog installs)	VERIFIED-LIVE
<code>hermes mcp serve</code>	Run Hermes AS an MCP server for other agents	VERIFIED-LIVE
<code>hermes computer-use install/status/doctor</code>	Desktop-control backend (cua-driver)	VERIFIED-LIVE
<code>hermes gateway run/install/start/status/setup</code>	Messaging gateway (Telegram, Discord, Slack, WhatsApp, ...)	VERIFIED-LIVE
<code>hermes send</code>	Send a message to a configured platform (scripts/cron/CI)	VERIFIED-LIVE

Automation & multi-agent

Command	What it does	Tag
<code>hermes cron list/create/edit/pause/resume/run/remove/doctor</code>	Scheduled jobs (durable, multi-platform delivery)	VERIFIED-LIVE
<code>hermes cron notepad</code>	Durable per-job key-value notepad across runs	VERIFIED-LIVE
<code>hermes kanban create/list/show/link/complete/swarm/...</code>	Durable multi-profile task board (40+ verbs)	VERIFIED-LIVE

Command	What it does	Tag
<code>hermes kanban swarm</code>	Generate a parallel-workers → verifier → synthesizer task graph	VERIFIED-LIVE
<code>hermes project create/list/add-folder/bind-board</code>	Named multi-folder workspaces (desktop Projects)	VERIFIED-LIVE
<code>hermes profile list/create/use/alias/export/import</code>	Isolated Hermes instances	VERIFIED-LIVE
<code>hermes auth add/list/priority/reset</code>	Pooled credentials per provider (rotation)	VERIFIED-LIVE
<code>hermes fallback list/add/remove</code>	Fallback model chain (auto-rollover on failure)	VERIFIED-LIVE
<code>hermes model</code>	Interactive model/provider picker	VERIFIED-LIVE

In-session slash commands (DOC-ONLY)

Source: <https://hermes-agent.nousresearch.com/docs/reference/slash-commands>

Command	What it does
<code>/help</code>	List all commands (authoritative in your version)
<code>/new (/reset)</code>	Fresh session
<code>/resume [name]</code>	Resume a named/recent session
<code>/branch (/fork)</code>	Branch the current session
<code>/compress</code>	Manually compress context (auto-compression also exists)
<code>/undo</code>	Remove last exchange
<code>/retry</code>	Resend last message
<code>/title [name]</code>	Name the session
<code>/save</code>	Save conversation to file
<code>/history</code>	Show conversation history
<code>/skill <name></code>	Load a skill into the session
<code>/skills</code>	Search/install skills
<code>/reload-skills</code>	Re-scan skill directory
<code>/tools / /toolsets</code>	Manage tools
<code>/goal [text]</code>	Set a standing goal the agent works toward across turns (<code>/goal status/pause/clear</code> to manage)
<code>/background <prompt></code>	Run a prompt in the background
<code>/queue <prompt></code>	Queue a prompt for the next turn
<code>/steer <prompt></code>	Inject a course-correction after the next tool call without interrupting
<code>/agents</code>	Show active agents and running tasks

Command	What it does
<code>/cron</code>	Manage cron jobs in-session
<code>/kanban</code>	Multi-profile collaboration board in-session
<code>/model [name]</code>	Show/change model mid-session
<code>/reasoning [level]</code>	Set reasoning effort
<code>/voice [on off tts]</code>	Voice mode
<code>/rollback [N]</code>	Restore filesystem checkpoint (needs <code>--checkpoints</code>)
<code>/usage</code>	Token usage
<code>/insights [days]</code>	Usage analytics
<code>/platforms</code>	Gateway platform status

Note: Hermes compresses automatically near the context limit; no manual threshold watch is needed the way Claude Code's `/context` grid is.