Hermes Agent Made My AI Assistant Useful Again

June 1, 2026 / Gavin Jackson

hermes openclaw ai agent self-hosted nous-research kimi codex

It finally happened. My OpenClaw gateway stopped responding.

Maybe it was an update. Maybe it was plugin rot — things had been getting flakier for weeks. Scheduled tasks that executed but never sent notifications. Cron jobs that produced "empty-heartbeat-file" errors and then went silent. Comms channels that were technically connected but somehow never delivered messages. My AI assistant — affectionately named Bob — had become very good at pretending to work while doing absolutely nothing.

I tried to bring the gateway back. I reinstalled it. Twice. I wiped the config and started over. I followed every troubleshooting thread I could find — split brain installs, protocol mismatches, invalid config guards, the whole command ladder. Nothing stuck. The unresponsive gateway had made the system unusable, and I was done fighting it.

I'd written about OpenClaw's beautiful mess before. The vision was always compelling: a persistent AI agent that lives in your environment, remembers what you teach it, and operates across multiple channels. But the execution felt permanently pre-beta. Like a brilliant proof of concept that needed another year in the oven.

So when OpenClaw finally ghosted me for good, I started looking for alternatives. And that's when I found **Hermes Agent** from Nous Research (MIT licensed Open Source project).

The Migration: Two Days, One Tiny VPS

I spun up a fresh BinaryLane VPS. Two gigabytes of RAM. That's it. One of the cheapest instances they offer. If Hermes could run here, it could run anywhere.

For context, I'd been running OpenClaw on a 2019 Intel MacBook Pro with 32 GB of RAM sitting on a pantry shelf. It had the horsepower, but it never felt like the bottleneck was hardware — it was the software.

Installation was suspiciously straightforward:

```
curl -fsSL https://hermes-agent.nousresearch.com/install.sh | bash
hermes setup
```

That was it. No dependency hell. No wrestling with Python environments. No gateway service that needed its own init script and log rotation and prayer. Just a single binary and a setup wizard.

I pointed it at **Kimi K2.6** through OpenRouter, added my **Brave Search API** key for web lookups, and configured a Telegram bot so I could talk to Bob from my phone. Within about twenty minutes, I had a working AI agent running on a $5-a-month instance.

Then I hooked up the integrations that matter to me:

- **GitHub**: Hermes can read, write, and manage issues across my private repositories. All my project tasks, blog ideas, and tracking issues live in GitHub now — version controlled, backed up, and accessible.
- **Obsidian**: My markdown notes vault at `~/Documents/obsidian-vault` is directly readable and writable by the agent. Journal entries, research notes, project documentation — all in plain text, all portable, all mine.
- **Codex CLI**: For actual coding work, I can delegate to OpenAI's Codex agent directly from within Hermes. It clones repos, writes code, runs tests, and reports back.

What Hermes Does Differently

The first thing that struck me was the memory architecture. OpenClaw had context windows that filled up and then... well, then your agent became amnesiac. You'd spend half a session re-explaining what you were working on.

Hermes takes a fundamentally different approach. It maintains three persistent memory files that survive across sessions:

- **USER.md**: Who I am, what I like, how I communicate. It learns my preferences and keeps them.
- **SOUL.md**: The agent's own operating model — how it approaches tasks, what skills it has, how it delegates.
- **MEMORY.md**: Facts, conventions, project structures, lessons learned. Durable knowledge that compounds over time.

After two days of testing, this isn't theoretical. I told Bob once that I prefer concise responses. That's in USER.md now. I explained the structure of my GitHub projects. That's in MEMORY.md. I showed him how I like my blog posts formatted. He remembered.

But the real differentiator is the **learning loop**. Hermes doesn't just store memories — it creates skills from experience. When I walk Bob through solving a problem, he can convert that workflow into a reusable skill. Next time a similar task comes up, he loads the skill, applies the prior steps, and refines them. That's not a chatbot. That's an agent that actually gets better the longer you run it.

The Feature Gap Is Real

Let's talk about what Hermes does that OpenClaw either couldn't do or couldn't do reliably:

Multi-channel presence that actually works. Hermes supports CLI, Telegram, Discord, Slack, WhatsApp, Signal, Email, Matrix, and more. I've only tested Telegram so far, but messages send and receive instantly. No silent failures. No "channel connected but messages not flowing." It just works.

Natural language cron scheduling. You literally type `every morning at 8am send me a summary of my calendar` and Hermes creates the cron job. I've got a morning briefing running that checks my schedule and sends me a Telegram message. It's been reliable for two days straight, which is already a better track record than OpenClaw managed in two months.

Subagent delegation. Hermes can spawn isolated subagents for parallel work. I can ask it to research something, write code, and draft an email — all simultaneously — then synthesise the results. Codex integration means the coding subagent is actually competent, not just hallucinating bash commands.

Open standard skills. Hermes uses the agentskills.io open standard for skill portability. Skills aren't locked into a proprietary format. That's a big deal for an open-source project.

Persistent memory across sessions. I can end a conversation, come back tomorrow, and Bob remembers where we left off. Not because the context window is huge — though Kimi K2.6 certainly helps — but because the important stuff is persisted to disk in a structured way.

Self-hosting that doesn't require a data centre. I'm running this on 2GB of RAM. Two. Gigabytes. OpenClaw needed a beefier machine just to keep the gateway and model context happy. Hermes feels lightweight by comparison.

A Security Reality Check

I should mention the security angle, because I initially thought Hermes might have a cleanliness advantage on the CVE front. It doesn't, at least not definitively. Public vulnerability indexes do list Hermes Agent CVEs, including issues around gateway authentication and file handling.

The honest take is this: both OpenClaw and Hermes are powerful automation tools with broad system access. You should firewall the gateway, keep them updated, run them on isolated infrastructure, and treat them like the privileged systems they are. Hermes being newer doesn't make it magically

immune to security issues.

For my own setup, I'm using Pangolin as the remote access layer — no open ports on the VPS, zero-trust authentication at the edge, and the Hermes gateway is only reachable through that tunnel.

What Hermes does offer is a more transparent architecture. The code is open source under the MIT license. The memory files are plain markdown you can read and edit. The skills are portable. There's no black box — you can see what it knows and how it thinks.

Two Days In

It's early days. I know that. Two days is not a longitudinal study. But here's what I can report so far:

- The gateway has not crashed once.
- Telegram messages deliver instantly in both directions.
- Cron jobs execute and actually notify me.
- Coding tasks delegated to Codex complete successfully and report back.
- GitHub integrations work for issue creation and repo management.
- Obsidian notes are readable and writable.
- The memory system is populating itself with useful context about who I am and how I work.

Most importantly: when I ask Bob to do something, he does it. He doesn't describe what he would do. He doesn't promise to update the task system and then silently fail. He executes.

That's the difference between a language model operating in "description mode" and an agent operating in "execution mode." OpenClaw often felt like the former. Hermes, so far, feels like the latter.

Should You Switch?

If you're running OpenClaw and it's working for you — genuinely working, not just technically alive — then maybe you don't need to switch. Migration costs time and energy.

But if you're fighting your agent more than you're using it… if cron jobs are imaginary… if the gateway needs regular exorcism… if you've ever watched your assistant confidently describe a fix without actually applying it… then Hermes is worth a look.

The install is trivial. The resource requirements are modest. The integrations are broad. And the architecture — persistent memory, skill creation, subagent delegation, multi-channel delivery — feels like what OpenClaw was always meant to become.

OpenClaw had the vision. Hermes, I think, has the execution.

I'll report back after a month of real use. But if the first two days are any indication, Bob has found a new home.

References

- Hermes Agent — Official Project Website
- Hermes Agent Documentation
- Hermes Agent on GitHub
- Nous Research
- agentskills.io — Open Standard for AI Agent Skills
- OpenClaw Troubleshooting Guide
- My earlier post: OpenClaw Is a Beautiful Mess
- My post on Pangolin for zero-trust remote access
- Kimi K2.6 on OpenRouter