What This Is

Hermes Agent is a free, open-source AI agent that runs on your computer. Think of it as an AI assistant that can actually *do* things — read and write files, run terminal commands, search the web, schedule recurring tasks, and talk to you through Telegram, Discord, or any messaging app you already use.

You pick the AI model (OpenAI, Anthropic, DeepSeek, local models — 20+ providers). Hermes handles the tools, memory, scheduling, and multi-platform delivery.

Website: https://hermes-agent.nousresearch.com/docs/
GitHub: https://github.com/NousResearch/hermes-agent

Most AI chats forget everything the moment you close the tab. Hermes doesn't. It has persistent memory, self-improving skills, scheduled autonomous tasks, and runs as a messaging gateway so the same agent reaches you on Telegram, Discord, Slack, WhatsApp, and more.

How to Install (macOS / Linux / WSL)

Open your terminal and paste:

```bash
curl -fsSL https://raw.githubusercontent.com/NousResearch/hermes-agent/main/scripts/install.sh | bash
```

Then run the setup wizard:

```bash
hermes setup
```

It walks you through picking a model, configuring API keys, and connecting messaging platforms.

How to Pay for the AI Model (Affordably)

You need an AI model to power the agent. Here are the cheapest practical paths:

Free Tier (zero cost, real models):

Provider	What You Get	Setup

OpenRouter free models	50 req/day across 20+ free models including Llama 3.3 70B, DeepSeek V4 Flash, Gemma 4 31B, Qwen3 Coder	Sign up at https://openrouter.ai → get API key → good enough to start
Google AI Studio	Gemini 3 Flash (250K tokens/min), Gemma models (14K req/day), free tier	https://aistudio.google.com → get API key
Groq	Fast open-model inference, generous free tier	https://console.groq.com
GitHub Copilot free tier	Included with GitHub account, works with Hermes	`hermes model` → GitHub Copilot → OAuth

Cheapest Paid (pennies per day for personal use):

Provider	Cost	Best Model for the Price
OpenRouter	Pay-as-you-go, no subscription	DeepSeek V3: $0.15/M input tokens, $0.60/M output — literally pennies per conversation
DeepSeek direct	~$0.14/M input, $0.28/M output	https://platform.deepseek.com
Together AI	$0.05–$1.00/M tokens, free credits to start	https://together.ai
Nous Portal	OAuth, no API key needed	https://portal.nousresearch.com

Free LLM resource list: https://github.com/cheahjs/free-llm-api-resources

Realistic cost: For personal daily use as an AI chief of staff, DeepSeek V3 through OpenRouter costs roughly $2–5/month. You don't need a $200/month ChatGPT Pro subscription.

Setup in Hermes after getting a key:

```bash
hermes model                  # interactive picker, or:
hermes config set model.provider openrouter
hermes config set model.default deepseek/deepseek-chat
```

Your First Commands

```bash
hermes                        # Start interactive chat
hermes chat -q "What is the capital of France?"    # Single question
hermes model                  # Change AI model
hermes doctor                 # Health check
```

Let Hermes Help You Learn Hermes

Here's the meta-trick: once installed, Hermes can configure and troubleshoot itself. Try:

```bash
hermes chat -q "Show me how to create a new Hermes profile for my research project"
hermes chat -q "Help me set up a cron job that summarizes my email daily"
hermes chat -q "I want to connect Hermes to Telegram. Walk me through it."
```

Hermes has a built-in skill (`hermes-agent`) containing its own documentation. Use the agent to learn the agent.

The Spec-First Mindset

Most people type random prompts and hope. The alternative:

Define what you want before you ask for it.

Instead of: "Hey can you help me organize my research?"

Try: "I want a system that checks my calendar every morning, pulls weather and news headlines, compiles a briefing note with a specific format, and delivers it to Telegram at 7 AM sharp."

That second version gives the agent enough to build something durable — a workflow, not a one-off answer.

Quick Spec Template

When asking Hermes to build something, include:

1. **Owner** — who or what owns this (a profile? a cron job?)
2. **Boundary** — what this does NOT do
3. **Inputs** — what data comes in
4. **Outputs** — what is produced
5. **Schema** — what format
6. **Schedule** — how often
7. **Verification** — how do we know it worked

Real Example: Personal Morning Briefing Assistant

Say you want a system that prepares a morning briefing for you every day — pulling in your calendar, weather, top news, and any unfinished tasks from yesterday. Spec-first answer:

1. Define the System

```
Owner: Dedicated Hermes profile called "briefing"
Boundary: Morning briefings only. No general chat, no file management, no code execution.
Inputs: Calendar events, weather API, news headlines, local task file
Outputs: A single daily briefing note delivered to Telegram at 7 AM
```

2. Define the Schema

Every briefing note follows the same format:

```
# Morning Briefing — Monday, May 25

## Weather
- Sunny, high of 78°F, 10% chance of rain

## Today's Calendar
- 9:00 AM — Team standup
- 2:00 PM — Client call
- 4:30 PM — Dentist appointment

## Top Stories
- [Headline 1 with link]
- [Headline 2 with link]
- [Headline 3 with link]

## Carry-Over From Yesterday
- Finish budget draft
- Reply to Sarah's email
```

3. Define the Schedule

- Daily at 6:30 AM: gather data from all sources, compile briefing, deliver to Telegram

4. Create the Profile

```bash
hermes profile create briefing --clone-from default --clone
```

Configure with only the tools it needs (web, cron, messaging, file access). Disable terminal, image gen, and anything else.

5. Verify

- Profile runs and can reach the weather API and calendar
- One briefing note is generated with correct format
- Briefing delivers to Telegram at the right time
- Dry run shows all data sources responding

The spec is the artifact. The code comes after.

Key Files and Paths

What	Where
Config	`~/.hermes/config.yaml`
API keys / secrets	`~/.hermes/.env`
Installed skills	`~/.hermes/skills/`
Session transcripts	`~/.hermes/sessions/`
Gateway / error logs	`~/.hermes/logs/`
Memory (durable facts)	`~/.hermes/memories/`
Cron jobs	`~/.hermes/cron/`
Profiles (isolated agents)	`~/.hermes/profiles/<name>/`
Source code (if cloned)	`~/.hermes/hermes-agent/`

Where to Learn More

- **Official docs:** https://hermes-agent.nousresearch.com/docs/
- **GitHub:** https://github.com/NousResearch/hermes-agent
- **Free LLM API list:** https://github.com/cheahjs/free-llm-api-resources
- **OpenRouter (cheapest model access):** https://openrouter.ai
- **Google AI Studio (free Gemini):** https://aistudio.google.com
- **DeepSeek (cheapest paid):** https://platform.deepseek.com
- **In-terminal help:** `hermes --help`
- **Ask Hermes itself:** `hermes chat -q "explain how cron jobs work"`

Start simple. Spec before code. Let the agent help you configure the agent.

Copy-Paste Prompts: Stand Up Your Own System

Use these inside Hermes chat or send them to Hermes via Telegram. Each prompt is self-contained — copy, paste, run. They build on each other in order, using a **morning briefing assistant** as the example. Swap in your own use case.

PROMPT 1 — Create a dedicated profile

```
Create a new Hermes profile called "briefing". Clone it from the default profile but only enable the tools this profile needs: web search, cron, messaging, file access, and terminal. Disable everything else (image generation, music, smart home, Discord, Spotify). Use OpenRouter with the DeepSeek V3 model to keep costs low. After creating it, run a smoke test to confirm the profile works.
```

PROMPT 2 — Define the workspace

```
Create a folder at ~/briefing-data/ with subfolders for: daily-briefings, templates, and logs. Inside templates, create a file called briefing-template.md with sections for Weather, Today's Calendar, Top Stories, and Carry-Over From Yesterday. Save this path to the briefing profile's memory so it persists.
```

PROMPT 3 — Define the output schema

```
Save this briefing format to the briefing profile's memory. Every daily briefing must follow this exact structure: a title line with the day and date, then four sections — Weather (conditions, temperature, precipitation), Today's Calendar (time, event name), Top Stories (up to 5 headlines with links), and Carry-Over From Yesterday (unfinished tasks from the previous day's task file). Remember this format permanently.
```

PROMPT 4 — Set up the data collection cron job

```
Create a cron job that runs daily at 6:30 AM. Its job: check my calendar for today's events, fetch the weather forecast for my location, pull the top 5 news headlines, and read my task file for any items marked as unfinished from yesterday. Compile everything into the briefing template format and save it to ~/briefing-data/daily-briefings/. Report what was gathered. If any data source fails, tell me which one and why.
```

PROMPT 5 — Set up the delivery job

```
Create a cron job that runs daily at 7:00 AM. Find today's briefing file in ~/briefing-data/daily-briefings/, format it as a clean message, and deliver it to this Telegram chat. If the briefing file doesn't exist or is empty, send an alert instead.
```

PROMPT 6 — Set up a weekly review job

```
Create a cron job that runs every Sunday at 7 PM. Its job: look through the past week's briefings, summarize the 3 most important events, list any recurring themes, flag tasks that were carried over multiple days, and suggest one thing to prioritize for the coming week. Deliver the review to this Telegram chat.
```

PROMPT 7 — Configure Telegram routing

```
I want the briefing profile to only answer in a specific Telegram topic. Set allowed_threads so it cannot respond in any other thread even if mentioned. Use require_mention: false so it responds to all messages in its assigned thread. Show me the exact config changes and what Telegram thread ID I need to provide.
```

PROMPT 8 — Test the full pipeline end to end

```
Run a dry-run test of the entire pipeline. Don't create or modify any files — just walk through what would happen: show me what calendar events would be found, what the weather would look like, what news headlines would appear, what the compiled briefing note would look like, and confirm all cron jobs are registered. Report any gaps or missing configuration.
```

PROMPT 9 — Learn to interact via Telegram

```
Once the briefing profile gateway is running, here's how to talk to it. In its assigned Telegram topic, send messages like: 'Show me today's briefing', 'What's on my calendar tomorrow?', 'Run the morning briefing now', 'Add a task to my carry-over list', 'Check the weather for this weekend', 'Summarize my week so far'. The profile only answers requests related to its job — it will ignore anything else.
```

PROMPT 10 — Ongoing maintenance commands

```
Teach me the commands for ongoing maintenance: check cron job status (hermes -p briefing cron list), run a specific job immediately (hermes -p briefing cron run <job_id>), view gateway logs if something goes wrong (tail ~/.hermes/profiles/briefing/logs/gateway.log), restart the gateway (hermes -p briefing gateway restart), and smoke-test the profile (hermes -p briefing chat -q "Return exactly: BRIEFING_OK" --toolsets safe).
```