BEGINNER FIELD MANUAL · 2026 EDITION

TechZeitGeist
Energie - KI- Mobilität

Hermes Agent
A Practical Beginner Guide

How Hermes works, how to install it, and how to use it from Desktop, CLI, TUI, Gateway, Docker and more.

Creator / Urheber: @TechZeitGeist - www.techzeitgeist.de

Created for an easy first week · Source base: official Hermes Agent docs

TechZeitGeist

Start here: the shortest path to a useful Hermes

If you are new, do not try to configure every feature on day one. Get one normal chat working first, then add tools, automation and integrations.

1. **Install** – Desktop installer or terminal one-liner
2. **Choose provider** – hermes setup --portal or hermes model
3. **First chat** – hermes, hermes --tui, or Desktop
4. **Add tools** – web, files, terminal, browser, voice
5. **Automate** – gateway, cron, delegation

The beginner recipe

1 If you use macOS or Windows

Download and run the Hermes Desktop installer. It gives you the app and the command-line tools, using the same config, sessions, skills and memory.

2 If you prefer the terminal

Install from the shell, run hermes model or hermes setup --portal, then start with hermes --tui or hermes.

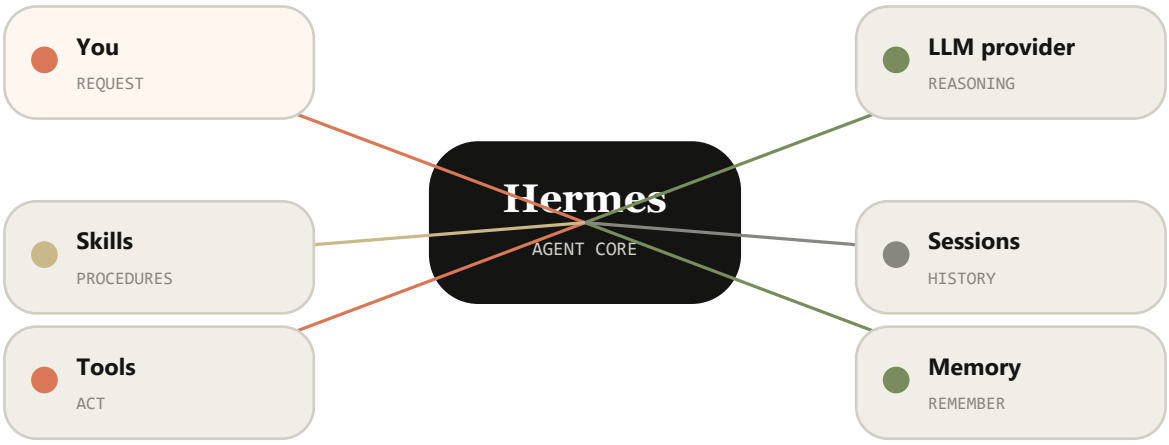
Copy-paste starter commands

```
# macOS / Linux / WSL2 / Android Termux command-line install
curl -fsSL https://hermes-agent.nousresearch.com/install.sh | bash

# Windows native install from PowerShell
iex (irm https://hermes-agent.nousresearch.com/install.ps1)

# Recommended first setup path for newcomers
hermes setup --portal
hermes --tui
```

Success looks like this: Hermes starts without provider errors, shows a model in the banner, can answer a second follow-up, and can use at least one tool such as file reading, terminal, or web search when you ask for it.

What Hermes Agent actually is

Think of Hermes as an AI agent runtime: a model for reasoning, tools for action, memory for continuity, skills for reusable know-how, and sessions so your work can resume later.



Hermes loops: understand → choose tool → observe result → continue → answer.

1	many	20+	local
AGENT CORE	FRONT ENDS	PROVIDERS	SESSIONS

The key idea: Hermes does not merely answer a prompt. It can decide that it needs a file, a shell command, a browser, a past conversation, a skill document or a subagent; call the right tool; read the result; and continue until it can give you a grounded answer.

That loop is why Hermes is useful for work: debugging a repository, researching a topic, creating a report, scheduling a reminder, running a browser task, or operating from a messaging app all use the same underlying agent.

Choose the interface that matches the moment

Hermes has several front ends. They share the same core, configuration, sessions, skills and memory, so you can move between them instead of starting over.

D Desktop App

Best first experience: chat, files, voice, settings, sessions.

T TUI

Recommended terminal interface: modal overlays, session switcher.

C Classic CLI

Fast, scriptable, familiar terminal chat for power users.

G Gateway

Telegram, Discord, Slack, Email, WhatsApp, Matrix and more.

Use Desktop when...

you want the easiest onboarding, visual settings, drag-and-drop files, chat previews, voice, and project navigation.

Use TUI when...

you live in a terminal but want a modern interface: modal model picker, session switcher, mouse selection and responsive overlays.

Use Classic CLI when...

you want speed, scripts, single-shot commands, SSH friendliness, or a minimal terminal session.

Use Gateway when...

you want Hermes in Telegram, Discord, Slack, WhatsApp, Signal, Email, Matrix, Home Assistant or other platforms.

```
Interface commands
hermes                  # classic interactive CLI
hermes --tui            # modern terminal UI, recommended for interactive
    terminal use
hermes chat -q "Summarize this folder in 5 bullets"
hermes --continue       # resume the most recent session
hermes desktop          # launch/install the desktop app after a CLI
    install
```

Installation paths by operating system

The official docs recommend the Desktop installer on macOS and Windows for the easiest start. The command-line installer remains the simple path for Linux, macOS, WSL2 and Termux.

PLATFORM	START HERE	BEGINNER ADVICE
Windows 10/11	`Desktop installer or PowerShell`	Use native unless you need POSIX-heavy workflows or the dashboard /chat PTY.
macOS	`Desktop installer or curl installe`	Great fit for app + terminal. Grant microphone access if you use voice.
Linux	`curl installer or Docker`	Best for servers, gateway bots, cron, and always-on automation.
WSL2	`standard Linux install inside WSL`	Pick it for Linux dev tools, POSIX paths, or dashboard embedded terminal.
Docker	`nousresearch/hermes-agent image`	Use for repeatable server deployments; mount ~/.hermes to keep data.
Android / Termux	`curl installer, Termux-specific pa`	Works for portable CLI use; voice extras have Android limitations.

What the installer handles: Hermes' installer provisions the project, virtual environment, the global `hermes` command and important dependencies such as Python, Node.js, ripgrep and ffmpeg where needed. After installation, reload your shell or open a new terminal.

Rule of thumb: native Windows is good for most users; WSL2 is better if your projects and developer tools already live in Linux paths or you need POSIX behavior. Docker is best when you want a clean server deployment or a repeatable environment.

Set up a model provider before anything else

Hermes needs at least one LLM provider. New users should avoid juggling many API keys at first; configure one provider, prove chat works, then expand.

N

Recommended newcomer path: Nous Portal

The official docs describe Nous Portal as a unified OAuth/subscription gateway covering 300+ models plus Tool Gateway features such as web search, image generation, text-to-speech and browser automation. One setup command can wire model and tools together.

```
Provider setup commands
hermes setup --portal       # OAuth + provider + Tool Gateway in one
    guided flow
hermes model                # choose or change provider/model
    interactively
hermes portal info          # inspect portal login and routing
hermes doctor               # check config and dependencies if something
    feels wrong
```

A

Other common choices

OpenRouter for broad model access, Anthropic or OpenAI-compatible APIs for direct billing, GitHub Copilot/Codex OAuth paths, Google Gemini, xAI, DeepSeek, MiniMax, Qwen, Kimi and others.

L

Local or self-hosted models

Use hermes model and select a custom endpoint such as Ollama, vLLM, llama.cpp or SGLang. Hermes expects a large enough context window for reliable tool use; the docs mention 64k tokens as the agent minimum for local setups.

Beginner warning: if a plain chat cannot complete normally, do not add gateway, cron, voice or complex routing yet. Fix the provider first. Most frustrating Hermes setups fail because the base model/provider was never verified.

Your first chat: ask for something verifiable

A good first prompt gives Hermes enough context and produces an answer you can check. Avoid vague tests like “do something cool.”

P

Use a concrete task

Good first prompts: “Check my current directory and tell me what this project is,” “Summarize this repo in five bullets,” or “Create a one-page plan for setting up my blog workflow.”

```
First chat exercise
hermes --tui
# then ask:
Check my current directory and tell me what looks like the main project
    file.

# resume after quitting
hermes --continue
hermes -c
```

W

How to write useful prompts

Tell Hermes the goal, the source of truth, the constraints, and what done means. Example: “Read README.md and package files, then tell me how to run tests. Verify by running the safest listed command.”

S

How to steer mid-work

If Hermes starts in the wrong direction, interrupt or steer it. In terminal sessions slash commands such as /stop, /retry, /undo, /model, /tools and /help are your control panel.

Practical habit: ask Hermes to verify work before reporting success. This matters for files, code, commands and generated documents. A reliable agent should show real execution or file output, not just a plan.

CLI and TUI survival kit

The terminal interfaces are powerful because they combine conversation, tools, history, slash commands, model switching and filesystem access in one place.

```
Launch patterns
hermes
hermes --tui
hermes chat -q "Hello"
hermes chat --provider nous
hermes chat --toolsets web,file,terminal
hermes -s github-pr-workflow
hermes --resume <session_id>
hermes -w
```

/

Important slash commands

/help shows commands. /model changes model inside a session. /tools manages toolsets. /skills searches/loads skills. /status shows session state. /agents shows running subagents. /background runs a separate prompt. /new starts fresh.

T

Classic CLI vs modern TUI

The classic CLI is a stable prompt_toolkit terminal interface with multiline editing and autocomplete. The modern TUI is the recommended interactive terminal front end, with modal overlays, mouse-friendly selection, live session switching and richer runtime panels.

Beginner keyboard tip: on Windows terminals, Ctrl+Enter is the reliable way to insert a newline in Hermes. Shift+Enter may not arrive as a distinct key in many terminals.

Desktop App: the friendly front door

The desktop app is not a separate lightweight clone. It uses the same agent core, settings, sessions, skills and memory as the terminal and gateway.

C — Chat

Streaming replies, live tool activity, queued prompts and shared session history.

F — Files

Browse project folders and attach files by drag-and-drop.

V — Voice

Speak to Hermes and hear replies when providers are configured.

S — Settings

Manage models, tools, MCP servers, gateway and credentials without editing YAML.

Use Desktop when handing Hermes to a non-developer or when you want visual confidence. Use the terminal when you need exact commands, SSH, scripts, or repo-local development workflows. You can switch because both surfaces share the same underlying sessions.

Tools and toolsets: what Hermes can do

Tools are functions Hermes can call. Toolsets are bundles that decide which tools are available in a session, platform or task.

F file

Read, search, write and patch files.

T terminal

Run shell commands and manage background processes.

W web/browser

Search, extract pages, automate browsers and forms.

V vision/image/voice

Analyze images, generate media, transcribe or speak text when providers are configured.

M memory/session

Remember durable preferences and search past conversations.

D delegation/cron

Run subagents and scheduled tasks.

```
Tool management
hermes tools                # interactive tool manager
hermes tools list           # show toolsets and status
hermes tools enable browser
hermes tools disable homeassistant
hermes chat --toolsets web,file,terminal
/tools                      # in-session tool management
```

Safety model: more tools means more capability and more responsibility. Give Hermes the tools it needs for the job, but do not enable unrelated high-impact tools everywhere. Destructive shell commands are guarded by approval prompts unless you deliberately bypass them.

Media, web and browser capabilities

Once the base chat works, Hermes can search, browse, see images, generate media, transcribe voice and speak replies — depending on the providers and toolsets you enable.

W Web search

Look up current information and extract webpages when the web toolset is enabled.

B Browser automation

Navigate sites, click buttons, fill forms and inspect pages with a local or cloud browser backend.

V Vision

Paste or attach screenshots and images; ask Hermes to explain, compare or extract visual details.

A Voice and TTS

Use microphone input, spoken replies and messaging voice notes when STT/TTS providers are configured.

I Image generation

Create images from prompts when an image provider such as FAL/Tool Gateway is configured.

G Tool Gateway

Nous Portal can bundle web, browser, image generation and TTS routing so new users avoid per-tool key juggling.

```
Media and browser commands
hermes setup --portal        # bundled model + Tool Gateway path
hermes tools enable browser
hermes tools enable vision
/browser                     # connect a browser when needed
/voice on                    # voice-to-voice mode
/image                       # attach an image in the CLI
```

Beginner advice: browser and media tools are powerful but add moving parts. Enable them after the core provider works, and test each tool with one simple request before using it in an automated workflow.

Configuration, sessions and profiles

Hermes keeps its practical life in a local Hermes home directory: configuration, secrets, auth, sessions, skills, memory and profiles.

1 **config.yaml**

Main settings: model, provider, tools, terminal backend, display, memory, gateway and more.

2 **.env / auth.json**

API keys, OAuth credentials and provider tokens. Treat these as secrets.

3 **sessions / state.db**

Conversation history and searchable session state so work can resume later.

4 **skills / memories**

Reusable procedures and durable facts that make Hermes fit your workflow.

5 **profiles**

Separate Hermes identities with isolated config, sessions, skills and memory.

```
State and profile commands
hermes config path           # locate config.yaml
hermes config env-path       # locate .env
hermes config edit           # edit configuration safely
hermes sessions list         # find previous sessions
hermes profile list          # list profiles
hermes profile create work   # create an isolated profile
```

Practical rule: use a separate profile when experimenting with models, risky tools, bots, or project-specific memories. It keeps the beginner setup clean.

Skills and memory: how Hermes improves over time

Beginners often mix these up. Memory stores durable facts. Skills store reusable procedures. Sessions store what happened in a conversation.

Memory

Use memory for stable facts that should survive sessions: your name, response preferences, recurring environment details, or durable conventions. Keep it compact and high-signal.

Skills

Use skills for workflows: how to review a PR, publish an article, operate a tool, debug a system, or follow a project-specific process. Skills are markdown procedure documents Hermes can load on demand.

What not to save

Do not put temporary task progress, one-off issue numbers, stale PR IDs, or raw data dumps into memory. If it will be stale next week, it is probably a session detail, not memory.

```
Skill commands
hermes skills list
hermes skills search github
hermes skills install <skill-id>
/skill hermes-agent          # load a skill inside a session
/skills                      # browse/search skills interactively
```

Practical pattern: after Hermes solves a tricky, repeatable task, ask it to save the method as a skill. Next time it can load that procedure instead of rediscovering the same steps.

Working with files, folders and projects

Hermes becomes much more useful when it can see the right context: files, folders, git diffs, project instructions and previous sessions.

C

Project context files

Hermes can load project instructions from files such as .hermes.md, HERMES.md, AGENTS.md, CLAUDE.md, SOUL.md and Cursor rules. These shape how it behaves in that workspace.

@

Context references

Use @ references to point Hermes at files, folders, diffs or URLs instead of pasting everything manually. This keeps prompts shorter and gives the agent source material.

R

Checkpoints and rollback

When enabled, Hermes can snapshot a working directory before file changes. If a change goes wrong, /rollback gives you a safety net.

P

Good project prompts

“Read the repo instructions, inspect the test setup, make the smallest safe change, run the relevant tests, and report exact commands and results.”

Beginner advice: let Hermes inspect before editing. A good agent reads the README, package files, test config and project instructions first. If it writes code before understanding the project, steer it back.

Extensions: MCP, API server, dashboard and plugins

These are not required on day one, but they explain why Hermes can grow from a local assistant into a connected work platform.

M MCP servers

Add external tool servers — for example browser, database, IDE or SaaS integrations — and expose their tools to Hermes.

A API server

Expose OpenAI-compatible or webhook-style surfaces so other apps can talk to Hermes through the gateway.

D Web dashboard

Use a browser-based admin surface for sessions, jobs, metrics, configuration and gateway management.

P Plugins

Extend providers, tools, memory backends, media generation, routing and lifecycle hooks.

```
Extension commands
hermes mcp list
hermes mcp add <name>        # add an MCP server
hermes mcp test <name>
hermes plugins list
hermes gateway run           # hosts gateway features and API surfaces
hermes dashboard             # where available/configured
```

Beginner filter: do not install every integration because it exists. Add one integration only when you can name the job it should perform and how you will verify that it worked.

Automation: when Hermes works beyond one chat

After normal chat works, Hermes can run as a background assistant, scheduled job runner, multi-platform bot or parallel subagent coordinator.

Gateway

A background process connects chat platforms and keeps sessions alive.

Cron

Scheduled prompts run later or repeatedly, optionally with skills and scripts.

Delegation

Subagents handle parallel research, coding, or review in isolated contexts.

```
Automation commands
hermes gateway setup         # configure Telegram, Discord, Slack, etc.
hermes gateway run           # foreground gateway process
hermes gateway install       # install as a background service where
    supported
hermes cron create '0 9 * * *'
hermes cron list
/background Research this and summarize when done
/agents                      # monitor subagents and running work
```

Start carefully: automation should come after a working provider, tool configuration and one verified normal chat. Scheduled or messaging-based agents can act while you are not watching, so keep prompts specific and permissions narrow.

Messaging Gateway: Hermes where you already talk

The gateway is a single background process that connects Hermes to messaging platforms, preserves sessions, handles cron jobs and can deliver voice messages.

`Telegram` `Discord` `Slack` `WhatsApp` `Signal` `Email` `Matrix` `Mattermost` `Home Assistant` `DingTalk` `Feishu/Lark` `WeCom` `Weixin` `Teams` `LINE` `ntfy`

G What it is good for

Team bots, personal assistants, mobile access, voice notes, platform-native file/image sharing, and scheduled reports delivered where people already read messages.

S What to configure first

Model provider, tool providers, platform token, user allowlist/admins, slash command permissions, and whether progress messages should stream or stay quiet.

```
Gateway lifecycle
hermes gateway setup
hermes gateway status
hermes gateway start
hermes gateway stop
hermes gateway restart
```

Security note: in group chats, Hermes is not automatically “your voice.” Configure who may talk to the bot and which slash commands non-admin users may run. Plain chat and admin slash commands are separate surfaces.

Which deployment style should I choose?

There is no single best way. Pick the least complicated setup that supports the work you actually want Hermes to do this week.

1 **I am new and on Windows/macOS**

Use the Desktop installer. Then try the app and hermes --tui.

2 **I live in Linux/macOS terminal**

Use the curl installer and start with hermes --tui.

3 **I use Windows but develop in Linux tools**

Use WSL2, store projects inside the Linux filesystem, and mind the Windows↔WSL boundary.

4 **I want an always-on bot/server**

Use Linux or Docker, configure gateway as a service, and mount persistent ~/.hermes data.

5 **I want local/private inference**

Use hermes model with Ollama/vLLM/llama.cpp/SGLang; verify context length and tool calling.

Windows reality check: Hermes runs natively on Windows 10/11. WSL2 is still useful for POSIX-heavy development, but it creates two worlds: Windows paths and Linux paths. Most confusion comes from not knowing which side a file or server lives on.

Safety, privacy and trust

Hermes is powerful because it can use tools. Treat that as real capability, not as ordinary chat. Give it access deliberately and verify external actions.

A Approvals

Dangerous shell commands can trigger approval prompts. Review before approving; avoid YOLO mode until you understand the risk.

S Secrets

Keep API keys in the configured secret stores/.env, not in prompts. Hermes has redaction features, but you should still avoid sharing secrets unnecessarily.

L Local state

Sessions, skills and memory are stored locally under the Hermes home/profile unless you configure external providers or gateways.

E External actions

Emails, posts, messages and public writes should be reviewed before sending. Ask Hermes to draft first; approve only when satisfied.

Beginner safety checklist: start with normal chat, enable only the tools you need, review file edits before committing, keep destructive commands behind approval, and use a separate profile for experiments if you are trying risky configurations.

```
Safety and diagnostics
hermes doctor              # health checks
hermes status --all        # component status
hermes config              # view config
hermes config edit         # edit YAML configuration
hermes profile create test # isolated config/sessions/skills/memory
```

Troubleshooting: fix the base layer first

When Hermes misbehaves, resist changing five things at once. Work from provider → session → tools → platform integration.

Hermes starts but cannot answer

Run hermes model, check credentials, then hermes doctor. Verify one plain chat before anything else.

A tool is missing

Run hermes tools or hermes tools list. Some tools need provider keys or a gateway subscription. Start a new session after tool changes.

Gateway bot is silent

Check hermes gateway status, platform token, allowlist/admin config, and gateway logs.

Local model acts strangely

Verify model name, OpenAI-compatible URL, and context length. Tool-using agents need a large enough context window.

Windows/WSL path confusion

Confirm whether Hermes is running native Windows or inside WSL2; paths and localhost may mean different machines.

Debugging habit: keep a small known-good test prompt. If that prompt works in the CLI but fails in Telegram or Discord, the model is fine and the problem is likely gateway configuration. If it fails everywhere, fix provider/model first.

Copy-paste prompt recipes

Good prompts tell Hermes the goal, the source of truth, the constraints, the verification step and the desired output format.

1 Project inspection

Read the project instructions, README and package/config files. Tell me what this project does, how to run it, and what you verified.

2 Safe file change

Inspect the relevant files first. Propose the smallest safe change, apply it, run the most relevant check, and show exact results.

3 Research brief

Research this topic with current sources. Separate facts from uncertainty, cite source URLs, and end with a beginner-friendly summary.

4 Automation setup

Create a scheduled/gateway workflow for this task. Keep permissions narrow, include the exact schedule, and explain how I can pause or remove it.

```
Prompt structure
Use this structure:
Goal: <what you want>
Context: <files, URL, app, platform, constraints>
Verification: <how Hermes should prove it worked>
Output: <bullets, table, patch, PDF, message draft, etc.>
```

One strong sentence: “Do the work, verify it with real tool output, and tell me exactly what changed.” That sentence prevents many half-finished agent runs.

A first-week learning plan

A gentle sequence for becoming productive without drowning in configuration.

1 Day 1

Install, configure one provider, run hermes --tui, and verify resume with hermes --continue.

2 Day 2

Ask Hermes to inspect a safe folder or project and explain what it found.

3 Day 3

Enable/check file and terminal tools; ask for a verified, non-destructive task.

4 Day 4

Learn slash commands: /help, /model, /tools, /skills, /status, /new.

5 Day 5

Try a skill-backed task and ask Hermes when a new skill would be worth saving.

6 Day 6

Set up one gateway platform or one cron job — not both at once.

7 Day 7

Create a reusable profile or project context file for your most common workflow.

Glossary: Provider = the LLM backend. Tool = an action Hermes can call. Toolset = a bundle of tools. Skill = reusable procedure. Memory = durable fact. Session = a conversation history. Gateway = messaging/background service. Profile = isolated Hermes configuration and data.

Source notes and official references

This guide is based on the official Hermes Agent documentation plus the installed Hermes Agent skill. For exact commands and newest details, prefer the docs pages below.

- https://hermes-agent.nousresearch.com/docs/getting-started/quickstart
- https://hermes-agent.nousresearch.com/docs/getting-started/installation
- https://hermes-agent.nousresearch.com/docs/user-guide/desktop
- https://hermes-agent.nousresearch.com/docs/user-guide/cli
- https://hermes-agent.nousresearch.com/docs/user-guide/tui
- https://hermes-agent.nousresearch.com/docs/user-guide/windows-native
- https://hermes-agent.nousresearch.com/docs/user-guide/windows-wsl-quickstart
- https://hermes-agent.nousresearch.com/docs/user-guide/docker
- https://hermes-agent.nousresearch.com/docs/user-guide/configuration
- https://hermes-agent.nousresearch.com/docs/user-guide/profiles
- https://hermes-agent.nousresearch.com/docs/integrations/providers
- https://hermes-agent.nousresearch.com/docs/user-guide/features/overview
- https://hermes-agent.nousresearch.com/docs/reference/toolsets-reference
- https://hermes-agent.nousresearch.com/docs/user-guide/features/browser
- https://hermes-agent.nousresearch.com/docs/user-guide/features/voice-mode
- https://hermes-agent.nousresearch.com/docs/user-guide/features/image-generation
- https://hermes-agent.nousresearch.com/docs/user-guide/features/mcp
- https://hermes-agent.nousresearch.com/docs/user-guide/features/api-server
- https://hermes-agent.nousresearch.com/docs/user-guide/features/web-dashboard
- https://hermes-agent.nousresearch.com/docs/user-guide/messaging/
- https://hermes-agent.nousresearch.com/docs/reference/slash-commands
- https://hermes-agent.nousresearch.com/docs/reference/faq

Produced by TechZeitGeist

Creator / Urheber: @TechZeitGeist - www.techzeitgeist.de

Hermes is best learned by doing one real, small job end-to-end: inspect context, act with tools, verify, then summarize. Once that feels boringly reliable, add the shiny parts — voice, gateway, cron, subagents and custom skills.