Edition · **July 2026**
Always current online

LEXINGTON ROAD PRESS

The *Hermes* Project

COMPANION GUIDE

The living companion to Chapter 16 — the version-specific setup steps that change too fast to print.

PRACTICAL AI

A Field Guide to Working with Claude · M. Emmett Townsend

NEWTON · IOWA · EST. 2026

LEXINGTONROADPRESS.COM/COMPANION

START HERE

Why this guide *exists*

The book holds the parts that last. This holds the parts that change.

Chapter 16 ends with an invitation: the **Hermes Project**, an optional capstone where you stop reading about AI agents and go meet one. The trouble is that the exact tools, prices, and setup steps for that hands-on work move faster than a printed page can keep up with — the agent landscape in the book's own snapshot will have shifted by the time you hold it. So the durable ideas stayed in the book, and the perishable specifics moved here, where they can be kept current.

That division is deliberate, and it is the same lesson the whole book teaches: know which things are stable and which are moving, and treat each accordingly. The four framework questions at the heart of the project have not changed and will not. The command to install a piece of software will. This guide carries the second kind and points you back to the book for the first.

THIS GUIDE UPDATES — THE BOOK DOES NOT

Everything here reflects the landscape as of **July 2026**. Product names, prices, and commands are checked against the live sources at the time of writing. When you sit down to do the project, confirm the specifics against the official links in the *Resources* section at the end — the verify habit from Chapter 2 is exactly the skill this page is built to require. The newest version of this guide always lives at **lexingtonroadpress.com/companion**.

ORIENTATION

The Hermes Project, *in brief*

Everything up to Chapter 16 you can read. This you can do — and doing it is the surest way to turn what you studied into something you understand.

The project asks you to meet a real AI agent: to watch one work, or steer one, or build one — and then to judge for yourself what you would and would not trust it to do on its own. It comes in three paths so that everyone can take part regardless of budget, hardware, or administrator access. **All three are full-credit paths.** Observe is not the lesser option; the judgment you practice is the same judgment whether you ran the agent or watched it carefully.

PATH 1 · OBSERVE

Watch a careful walkthrough of an agent doing a real multi-step task, and answer the four questions about what you saw. No account, no spending, no setup.

PATH 2 · BUILD

With an adult's help where needed, set up a real open-source agent, connect it to a model with an API key, give it a small bounded task, and watch it work.

PATH 3 · EXTEND

Take a built agent further — a harder task, a new capability, or push it until it fails — and write a short case study of where your judgment had to step in.

THE SETUP IS NOT THE ASSIGNMENT

Read this next line twice. The installing and the clicking are not what you are being graded on — the **four framework questions are**. They are this whole book, asked about one agent. Whichever path you take, the questions are the work; the setup is only how you get something worth answering them about.

BEFORE YOU START

The one rule: *you are the friction*

An agent's whole value is that it runs without you watching every step. That is also its whole danger.

Chapter 16 named three tensions that shape every agent: capability versus trust, autonomy versus oversight, cost versus value. You are about to feel all three with your own hands. The more an agent can do, the more you have to trust it with; the moment it runs unwatched, you have given up the chance to catch it mid-mistake; and every capable model running in a loop is spending real money while it works.

So carry one rule into everything below: **you are the human in the loop, and the loop has to be real.** A real loop means you can actually catch and reverse a decision before it lands — not that the word "supervised" appears somewhere. Keep every task small enough, and bounded enough, that you can see what the agent did and undo it if it went wrong. Where this guide tells you to give an agent access to something, give it the least it needs and nothing you could not afford to lose.

IF YOU ARE UNDER 18 — OR SETTING THIS UP FOR SOMEONE WHO IS

Paths 2 and 3 involve creating an account, entering a payment method, and running software that can take actions on a computer. Do these with a parent, teacher, or another trusted adult, on a machine you are allowed to use that way. **Path 1 requires none of that** and earns the same credit — if any of the above is a barrier, take Path 1 and lose nothing.

PATH 1 · OBSERVE

Watch an agent *work*

No account. No cost. No installation. The same judgment, full credit.

The goal of this path is not to run an agent but to *read* one — to watch a real multi-step task unfold and notice where a human's judgment mattered. You are practicing the exact skill the whole project is about, without needing a key, a card, or a spare computer.

What to do

1. Find a careful, honest walkthrough of an AI agent doing a real, multi-step job from start to finish — booking something, researching and comparing options, working across files, negotiating, writing and revising code. A good walkthrough shows the *whole* run, including the moments the agent paused, asked, or got something wrong — not a polished highlight reel.
2. Watch it once all the way through, just to see the shape of it.
3. Watch it again with the four framework questions in front of you (see that section), stopping to note the exact moments where a person's judgment did — or should have — stepped in.
4. Write your answers. That is the assignment.

WHAT MAKES A GOOD SOURCE TO WATCH

- It shows a task with several dependent steps, not a single question and answer.
- It shows the failures and the pauses, not only the wins. The mistakes are where the lesson is.
- You can tell what the agent decided on its own versus what a human approved.
- Be skeptical of anything that looks like an advertisement. A demo never shows you the bill — or the times it broke. Apply Chapter 2's verify habit to the walkthrough itself: who made this, and what are they selling?

The book's own mid-2026 snapshot gives you two real cases to look for and compare: the user who let an agent spend days negotiating thousands of dollars off a car, and the security flaw (catalogued as **CVE-2026-25253**) that let a malicious website hijack an agent's access. Same class of tool, both stories true. If you can find honest write-ups of each, watching one against the other is the whole capability-versus-trust tension in a single sitting.

PATH 2 · BUILD

Set up a real agent and *give it a job*

You will connect a model to an open-source agent, hand it one small task, and keep your hand on the wheel while it works.

WHAT YOU NEED · ROUGHLY 30–45 MINUTES

- A computer you are allowed to install software on (Windows, macOS, or Linux), and an adult's help if you are under 18.
- A **Claude API key** — the programmatic door to the model, from Chapter 9. This is separate from a claude.ai chat account.
- A payment method and a small amount of credit (a few dollars is plenty for this project).
- An **open-source agent** to run on your own machine. This guide uses **OpenClaw**, the model-agnostic, bring-your-own-key personal agent from the book's snapshot.

Step A · Get a Claude API key

1. Go to **console.anthropic.com** and create an account. Even if you already use Claude Pro or the free app, API access is a *separate* account — the chat login will not work here.
2. Add a payment method and a small amount of credit. You cannot create a working key until billing is set up; a **$5** minimum is the fastest path and far more than this project needs.
3. In the left sidebar open **Settings → API keys**, then click **Create Key**. Give it a name like `hermes-project`.
4. Copy the key *now* and keep it somewhere private. It starts with `sk-ant-` and is shown only once — Anthropic does not store the secret, so if you lose it you simply make a new one.

TREAT THE KEY LIKE A HOUSE KEY

Anyone who has your API key can spend your credit. Never paste it into a chat, an email, a screenshot, or a public repository. If you ever think it leaked, delete it in the console and create a new one — that instantly revokes the old one.

Step B · Install the agent

OpenClaw runs on your own machine and lets you bring your own key, so you stay in control of both the model and the spending. You need **Node.js 22.14 or newer** (Node 24 is the safest choice) and **Git**. The fastest install is one command:

```
# install OpenClaw globally
npm install -g openclaw@latest

# run the guided setup wizard
openclaw onboard --install-daemon
```

The wizard walks you through the rest: it asks which model provider you want (choose **Claude**), where to paste your `sk-ant-` key, and which channels — if any — to connect. **For this project, keep it minimal.** You do not need to wire it into your messaging apps to learn the lesson; a plain local setup is safer and enough. If you would rather not install it directly on your system, OpenClaw also runs in Docker (`docker pull openclaw/openclaw:latest`), which keeps it walled off in a container.

PICK THE CHEAP MODEL — A BOUNDED TASK COSTS PENNIES

You choose which Claude model the agent uses. For a small learning task, choose the cheapest — **Claude Haiku 4.5** — and you will spend a few cents, not dollars. As of July 2026, per million tokens:

Haiku 4.5 — $1 in / $5 out · **Sonnet 5** — $2 in / $10 out · **Opus 4.8** — $5 in / $25 out. A short, bounded agent run uses only a sliver of a million tokens. Confirm current prices at the pricing link in Resources — they change.

Step C · Give it one small, bounded task — and watch

Do not point a brand-new agent at anything that matters. Pick a task that is real enough to be interesting but small enough that you can see every step and undo it:

- Research a narrow question across a few sources and produce a short, cited summary.
- Organize or rename the files in a single practice folder you made on purpose (never your real documents).
- Draft, run, and fix a tiny program to solve one small problem, in a fresh empty folder.

Then watch it work. When it asks permission to do something, *read the request before you approve it* — that pause is the entire point. Notice where it decides its own next step versus where it waits for you. Keep a running note of the moments that answer the four questions, because those moments are your assignment.

BOUNDING A TASK MEANS

- Work in a folder you created for this and would not miss — never your real files, photos, or documents.
- Do not connect it to email, banking, shopping, or anything that can spend money or send messages as you.
- Set a spending limit in the console so a runaway loop cannot run up a bill.
- Stay at the keyboard the whole time. If it starts doing something you did not intend, stop it.

PATH 3 · EXTEND

Push it until it *teaches you something*

Take a working agent past the easy demo and write down exactly where your judgment had to step in.

This path starts where Path 2 ends: you already have an agent running. Now you make it harder, and you pay close attention to where it breaks — because the breaking is the lesson. Pick one of these and go:

Give it a harder task

Move from a single clean job to one with more steps, more ambiguity, or a decision the agent cannot make on facts alone. Watch where it guesses, where it overreaches, and where it quietly does the wrong thing confidently.

Add a capability

Connect one new tool or data source — the Model Context Protocol (MCP) from Chapter 9 is the standard way agents plug into outside tools, and OpenClaw supports connectors. Add exactly one, give it a task that needs it, and notice how each new capability raises both what the agent can do *and* what it could get wrong.

Push it until it fails

Deliberately hand it something at the edge of what it can do and document the failure honestly. A clear account of *how* an agent breaks is worth more than a highlight reel of it succeeding.

THE DELIVERABLE · A SHORT CASE STUDY

Write one to two pages: what you asked, what the agent did, **where it helped, where it broke, and where your judgment had to step in.** Structure it around the four framework questions. The honest failures are the most valuable part — they are the friction the book has been about the whole way through.

THE ACTUAL ASSIGNMENT

The four *framework questions*

Answer these whichever path you took. They — not the setup — are the work. They are this whole book, asked about one agent.

1 · Judgment. Where in the task did human judgment matter most — where would the agent have gone wrong, or did it, without a person to catch it?

2 · Honesty and Scrutiny. What did the agent claim or assume that you had to verify? Where did you have to ask, "How do I know this is right?"

3 · Friction. Where was the human friction in this process, and what would have happened without it? If you removed yourself entirely, what breaks?

4 · The Road Ahead. Having watched this agent work, what would you trust an agent like it to do on its own, and what would you insist on keeping a human in the loop for — a real loop, not a labeled one?

A good answer is specific. Point to the exact moment — "when it assumed the second search result was current and it was two years old" — not a general impression. The difference between a labeled loop and a real loop (Chapter 16) is the sharpest lens you have for question 4: did the process let a human actually catch and reverse a decision in time, or did it only *print the word* supervised on a step where you structurally could not?

WHEN IT BREAKS

Troubleshooting

Almost every first-time snag is one of these. None of them mean you did something wrong.

"Invalid x-api-key" or an authentication error

The key is mistyped, has extra spaces, or was revoked. Copy it fresh from `console.anthropic.com → Settings → API keys`. If you lost it, just create a new one — keys are shown only once and are cheap to replace.

"Your credit balance is too low" / billing errors

A key exists but billing is not set up, or the credit ran out. Add a payment method and a few dollars of credit in the console. Remember API access is billed separately from any Claude Pro or Max subscription — paying for the app does not fund the API.

The install command fails, or "unsupported Node version"

OpenClaw needs Node.js 22.14+ (24 recommended). Check with `node --version`; if it is older or missing, install a current Node from nodejs.org and try again. On macOS/Linux you may need to prefix the global install per your setup; if permissions block it, the Docker install sidesteps this entirely.

The agent keeps asking permission for everything

That is not a bug — it is the loop working. Read each request and approve deliberately. If it is genuinely too chatty for a trivial task, narrow the task rather than turning off the prompts; the prompts are your safety.

The agent loops, stalls, or "runs away"

Stop it (close the run or interrupt the process). Give it a smaller, more sharply defined task — most runaways come from a goal that was too vague. This is worth noticing for the assignment: a vague goal plus autonomy is exactly how agents go wrong.

It is costing more than expected

Switch to **Haiku 4.5**, shorten the task, and set a spending limit in the console. Watching cost climb with autonomy *is* the cost-versus-value tension from Chapter 16 — note it for question 3.

> THE MOVE THAT FIXES MOST PROBLEMS
>
> When something set up wrong, go to the source, not a forum guess. The official docs (see Resources) are current; a printed guide — even this one — is a snapshot. The verify habit from Chapter 2 is the actual debugging skill: confirm against the authority before you rely on it.

DO NOT SKIP THIS

Safety & *permissions*

The same software that negotiated thousands off a car also carried a serious security flaw. Hold both.

Chapter 16 used one tool to tell both stories on purpose. On the capability side, an agent ran for days and saved a real person real money. On the trust side, security researchers found a vulnerability (**CVE-2026-25253**) that could let a malicious website hijack the agent's access and run commands on the user's own machine. Same class of tool, same week of headlines. An honest setup carries both at once — which is why these are rules, not suggestions.

- **Give the least access that works.** An agent can only do damage through what you connect it to. For this project, connect almost nothing — a single practice folder is enough.
- **Never connect money or identity.** No banking, shopping, email-as-you, or social accounts. Nothing that can spend or send on your behalf.
- **Guard the key.** It is spending power. Keep it private, set a spending limit, and revoke-and-replace at the first sign of a leak.
- **Beware what the agent reads.** The CVE above worked because an agent acted on instructions hidden in content it fetched. Do not point an agent at untrusted web pages or files and let it act unsupervised on what it finds there.
- **Keep a real loop.** Approve actions deliberately; do not click "approve" on autopilot. That reflex — consent fatigue, from Chapter 9 — is the human weak point that makes a powerful tool risky.
- **Under-18s work with an adult** for anything involving accounts, payment, or software that takes actions.

THE TEST THAT CARRIES THE MOST WEIGHT

Before you let an agent act on anything: *what would I need to check before I act on this, and who benefits if I simply believe it?* If you cannot answer, do not connect it. That one question is the whole book, and it is the resource that will still serve you when every product name in this guide has changed.

FOR INSTRUCTORS

Notes for *teaching it*

The project is built so every student can complete it fully — no student needs money, hardware, or admin access to earn full credit.

The three paths exist for equity, not for tiering. **Path 1 (Observe) is a full-credit path** and the right default for most classrooms: it needs no account, no spending, and no installation, yet it exercises the identical judgment. Reserve Paths 2 and 3 for students who have a suitable machine, adult support, and genuine interest — and make clear that choosing Observe costs nothing in grade or standing.

Grade the questions, not the setup

The assessment is the four framework questions, answered with specificity, regardless of path. A student who watched an agent carefully and pointed to exact moments of judgment has done the assignment more completely than one who installed everything and answered in generalities.

Framework question	What a strong answer shows
Judgment	Names a specific moment the agent would have erred without a person, not a general statement that "humans are important."
Honesty & Scrutiny	Identifies a concrete claim or assumption the student actually checked, and how.
Friction	Explains what specifically breaks if the human is removed — traced to a real step, not hypothetical.
The Road Ahead	Draws a defensible line between trust-on-its-own and keep-a-human, and applies the real-loop-vs-labeled-loop test to justify it.

Running it safely as a class

- Default the class to Observe; treat Build/Extend as opt-in with a consent step for families of minors.
- If students build, use a shared demonstration setup or a supervised lab rather than asking each student to enter a payment method.
- One well-chosen walkthrough shown to the whole class can anchor a rich discussion of all four questions with zero setup — and pairs naturally with the capability-versus-trust cases from Chapter 16.
- Emphasize the safety rules on the previous page before any hands-on work.

FURTHER READING · REFRESHED

Resources

The categories are durable; the specifics are perishable. Where an address is given, check that it still works — the web moves faster than any printed page, and when something has moved, the verify habit from Chapter 2 is what finds it.

Straight from the source

Claude documentation — docs.claude.com · The authoritative reference for capabilities, models, prompt-engineering, and release notes. When this guide and the live tool disagree, the documentation is right and the guide is the older snapshot.

Claude support — support.claude.com · Plain help for accounts, plans, settings, and step-by-step features — the "how do I actually do this" place.

Anthropic research — anthropic.com/research · The published work behind Claude: safety, capabilities, and Constitutional AI.

Anthropic news — anthropic.com/news · Official announcements. Read them with the five-question checklist from Chapter 16 and this page becomes practice, not just news.

For the Build path

Anthropic Console — console.anthropic.com · Where you create the API account, add credit, and generate your `sk-ant-` key. Separate from claude.ai.

API pricing — platform.claude.com/docs/en/about-claude/pricing · Confirm current per-token rates here before you assume this guide's numbers still hold.

OpenClaw — github.com/openclaw/openclaw · docs at docs.openclaw.ai · The open-source, model-agnostic, bring-your-own-key personal agent used in Path 2. Install and setup steps live here and stay current.

Model Context Protocol (MCP) — search "Anthropic MCP" for the current documentation and connector registry. The open standard that lets an agent connect to outside tools (Chapter 9), and the doorway for Path 3's "add a capability."

Where Claude's values come from

Claude's Constitution — anthropic.com/constitution · The written principles Claude is trained against (Chapter 1), plus a free audiobook narrated by two of its authors. Reading a few pages makes "training against principles" concrete.

The most important resource is not on this list, because it is a habit, not a link: whatever an agent produces, the person who uses it is responsible for checking what matters before acting on it. That habit is the whole point of the book — and the one resource that will still serve you when every product name on this page has changed.

The book holds the parts that last. This guide holds the parts that change.