

GUIDE

Second Brain Vault AI Handbook

12 8 7

SKILLS AGENTS DAYS TO FLUENCY

Your vault, your Claude, and the twelve skills that connect them – setup, first week, and the workflows that stick.

PART

01

Welcome: What You Just Bought

You now own two things people usually buy separately and never quite connect: a working knowledge system and an AI that knows how to run it.

You now own two things people usually buy separately and never quite connect: a working knowledge system and an AI that knows how to run it.

What's in the box

- **The vault.** 500+ notes in a structure refined by years of daily use: a home dashboard, an atomic note index with maturity stages ( seedling →  growing →  mature), a Library for books, Projects with Gantt views, task notes, and a review chain that runs from daily diary to yearly review. 298 notes are demo content, tagged `#aiassist`, so dashboards have something to show on day one.
- **12 Claude skills.** Ready-made procedures for the work you'll actually repeat: capture a thought, triage the inbox, write an atomic note, link it, review the day, plan the week.
- **8 agents.** Specialists for bigger jobs – a morning briefing, a deep weekly analyst, a three-voice board of advisors. Included as a bonus.
- **CLAUDE.md.** The rulebook. It tells Claude what every folder is for, which formats are load-bearing, and what it must never do: delete notes, touch your diary, invent data.
- **MCP configs** for Claude Desktop, plus a `.claude/` setup for Claude Code.
- **This handbook.** Eight parts, about forty pages.

What "AI layer" actually means

Picture a colleague who has read every note you've written, keeps a copy of your filing rules on their desk, and asks before making bulk changes. That's the whole idea. Claude isn't bolted onto the vault as a chat window – it opens each session by reading CLAUDE.md, so it knows that `Card Index/` holds one thought per note, that a wrongly formatted date silently breaks a database view, and that your diary is private. The skills give it repeatable procedures; the agents give it depth for the big jobs.

No plugin magic, no background sync, no hidden services. Your notes are plain files on your disk, the rules are plain text, and nothing leaves your machine except what you send to Claude in a conversation.

TIP

The demo notes are training wheels
Everything tagged `#aiassist` is safe to overwrite or archive –
Part 3 explains how the demo layer works.

Pick your path

- **New to Obsidian, Claude, or both** → go straight to **Part 2**. Setup takes about twenty minutes.
- **You already have a vault you love** → skip to **Part 7**. It covers what to back up, what to move, and in which order.
- **You just want the workflows** → **Parts 4 and 5**: a seven-day onboarding plan, then all twelve skills one by one.

Whichever path you take, start small. One captured idea today beats a perfect system next month.

PART

02

Setup: Claude Code and Claude Desktop

There are two ways to connect Claude to this vault. Claude Code is Anthropic's terminal agent – step into the vault folder and everything ships ready: instructions, 12 skills, guar...

There are two ways to connect Claude to this vault. **Claude Code** is Anthropic's terminal agent – step into the vault folder and everything ships ready: instructions, 12 skills, guardrails. **Claude Desktop** is the chat app – it reaches the vault through one small config file. Both take under ten minutes, and they can coexist. If you set up only one, make it Claude Code – the last section of this part explains why.

Before you start

Three things, then you're ready:

- 01 Put the vault somewhere permanent.** Unzip it to its final home – the Desktop config points at an absolute path, so moving the vault later means updating the config.
- 02 Check Node.js.** Claude Code installs through npm, and Desktop launches its file server through `npx`. Run `node --version`; 18 or higher is fine. If the command isn't found, install Node from nodejs.org first.
- 03 Have a Claude account.** A **Claude Pro subscription (\$20/month)** covers everything in this handbook. Both paths sign in with your normal Claude account – no API key, no per-token billing, nothing to top up. The free tier works for a first look but you'll hit its limits within a day; Max matters only for long daily agent sessions.

Path 1 – Claude Code (recommended)

Claude Code lives in the terminal. No config file – it works with folders directly, so you just start it in the right place.

Install it once:

```
npm install -g @anthropic-ai/claude-code
```

Then, every time you want to work with the vault:

```
cd "/path/to/Second Brain Vault"  
claude
```

On first launch, run `/login` – a browser window opens, you approve, done.

The `cd` line is the whole trick. Claude Code treats the folder you start it in as the project, and this vault is built for that. From the vault root, three things happen automatically:

- It reads `CLAUDE.md` – the folder map, the frontmatter contract, the working rules. That file is why an atomic note goes in `Card Index/` with a `maturity` field and not somewhere random.
- It discovers all 12 skills in `.claude/skills/`. Nothing to install or register – say "capture this" and the capture skill runs.
- It applies `.claude/settings.json` – the shipped permission list: destructive commands blocked, writes outside the vault denied.

Start Claude Code in any other folder and none of this happens. If Claude ever seems to have forgotten what the vault is, check where you launched it.

TIP

Make it a one-word habit

Add an alias to your shell profile – `alias vault='cd "/path/to/Second Brain Vault" && claude'` – and starting a correctly wired session becomes a single word. You'll never launch from the wrong folder again.

Path 2 – Claude Desktop

Claude Desktop can't see your disk on its own. It needs an MCP server – a small local program that hands Claude read and write access to one folder. The official filesystem server is all you need, and the vault ships a ready config for it: `Claude/claude_desktop_config.example.json`.

Four steps:

01 Find the Desktop config file:

- macOS: `~/Library/Application Support/Claude/claude_desktop_config.json`
- Windows: `%APPDATA%\Claude\claude_desktop_config.json`

If it doesn't exist yet, open Claude Desktop → Settings → Developer → Edit Config to create it.

01 Copy the example in. Paste the contents of `Claude/`

`claude_desktop_config.example.json` into that file (if you already use other MCP servers, merge the entry into your existing `mcpServers` block):

```
json { "mcpServers": { "second-brain-vault": { "command": "npx", "args": [ "-y",
"@modelcontextprotocol/server-filesystem", "/ABSOLUTE/PATH/TO/Second Brain Vault" ] } } }
```

01 Replace the placeholder path with your real one. It must be **absolute** – no `~`, no relative paths; both fail silently on some setups. On Windows, double the backslashes – `C:\\Users\\anna\\Documents\\Second Brain Vault` – JSON requires them.

02 Restart Claude Desktop completely. Quit the app – Cmd+Q on macOS, quit from the system tray on Windows – and reopen it. Closing the window is not enough: the config is read only at launch, and a half-restart is the most common reason this setup "doesn't work."

After the restart, open a new chat and look for the tools indicator – `second-brain-vault` should be listed. The first time Claude touches a file, Desktop asks for permission; approve it.

Verify the connection

Same test for both paths. Ask: **"What folders are in my vault?"** A correct answer lists `_Inbox`, `Card Index`, `Library`, `Projects`, `Tasks`, `Periodic Notes`, and the rest. Then one more: **"Read CLAUDE.md and tell me where a quick capture goes."** If Claude answers "a `#💡` line in `_Inbox/Ideas.md`," the whole loop is wired – it can see the files and it has read the rules.

If the first question fails: in Claude Code, you launched from the wrong folder – quit, `cd` into the vault root, relaunch. In Desktop, the path isn't absolute, the restart wasn't a full quit, or Node.js is missing.

Skills: where the two paths differ

Here's the honest part. **Automatic skill discovery works only in Claude Code.** Claude Desktop doesn't scan `.claude/skills/` – saying "capture this" in Desktop won't trigger the capture skill by name.

The skills themselves are plain Markdown playbooks, and Desktop can follow them as prompts: open `.claude/skills/capture/SKILL.md`, paste it into a conversation, and Claude executes the same algorithm through the filesystem server. It works; it's just manual. You are the discovery mechanism.

The practical split: keep Desktop open for quick questions, summaries, and reading sessions over the vault; the full experience – skills firing from a trigger phrase, the eight agents, the permission guardrails – is Claude Code. Run your first week (Part 4) there.

PART

03

How Claude Sees This Vault

Part 2 wired the connection; this part covers what Claude does with it. The first thing it reads each session is `CLAUDE.md` at the vault root – its map, its contract, and its house ...

Part 2 wired the connection; this part covers what Claude does with it. The first thing it reads each session is `CLAUDE.md` at the vault root – its map, its contract, and its house rules. Here is the same material from the human side: what Claude assumes about your folders, where it starts looking, and which formats are untouchable.

The folder map, through Claude's eyes

To you, folders are places. To Claude, they are type declarations. A file's location tells Claude what kind of note it is before it reads a single line.

FOLDER	WHAT CLAUDE ASSUMES
<code>_Inbox/</code>	Raw captures. Nothing here is final. <code>Ideas.md</code> collects one-line thoughts.
<code>Card Index/</code>	Atomic notes – one thought per file, tagged <code>#zettel</code> , each with a maturity stage.
<code>Library/</code>	Book summaries from the Book Search plugin, plus <code>Movies and Series/</code> and <code>University/</code> .
<code>Projects/</code>	One note per project, each with a Gantt block that pulls in linked tasks.
<code>Tasks/Tasks/</code>	TaskNotes files – one task per file, with machine-readable metadata.
<code>Periodic Notes/</code>	The reflection chain: Diary → Weekly → Monthly → Quarterly → Yearly.
<code>Bases/</code>	<code>.base</code> database files and their data folders.
<code>Templates/</code>	Blueprints for every note type. Claude reads them; it never edits them.
<code>Learning Obsidian/</code>	Reference guides, including the Vault Map and Tag Dictionary.
<code>Archive/</code>	Where outdated notes go. Claude moves things here instead of deleting. Always.

One subtlety about `Templates/`: those files contain Templater code like `<% tp.date.now("YYYYMMDD HH:mm") %>`, and that code only runs inside

Obsidian. So Claude copies a template's structure and fills in real values itself – raw `<% %>` code never lands in a note.

Entry points: !Home and the 00. hubs

When Claude needs orientation, it doesn't crawl five hundred files. It starts where you would:

- `!Home.md` – the dashboard that opens on vault startup. Recent notes, active tasks, today's entry – the state of the vault in one screen.
- **The `00. *` hubs** – `00. Zettelkasten`, `00. Reading`, `00. Habits`, `00. Finance`, and friends. Each hub is a topic map that links down into `Card Index/`. Ask about a topic, and Claude checks the matching hub first.
- `Learning Obsidian/Vault Map.md` and `Tag Dictionary.md` – the full folder tour and the tag glossary.

TIP

The same route works for you

Lost in the vault? Open `!Home` or the nearest `00. *` hub and follow the links – dashboards first, hubs second, full-text search last. That is exactly the path Claude takes.

The frontmatter contract

Here is the part that looks pedantic and isn't. Every note type has a fixed frontmatter shape, and the vault's Bases and Dataview blocks parse those fields literally. The Card Index base filters on `maturity == "🌱 seedling"` – the whole string, emoji included. Type `seedling` without the emoji and the note silently vanishes from every maturity view. No error message. Just gone.

NOTE TYPE	LIVES IN	THE LOAD-BEARING FIELDS
Atomic note	Card Index/	tags: [zettel], maturity (exact values below), created: YYYYMMDD HH:mm
Diary entry	Periodic Notes/ Diary/	tags: [diary] plus 11 numeric metrics: focus, learn, routine, move, connect, rest, sleep_hours, sleep_quality, energy, mood, stress
Weekly review	Periodic Notes/ Weekly Reviews/	tags: [weekly_review, diary], reviewed:
Book	Library/	Capitalized Book Search keys: Status, Title, Author, Pages, Cover...
Project	Projects/	tags: [project], status, priority, start_date, end_date, goals
Task	Tasks/Tasks/	status, priority, due, projects: "[[Project Name]]", dependsOn
Quick capture	_Inbox/Ideas.md	One line: - #💡 your idea - 20260712

Every field feeds something downstream. The `created` stamp – like the Added/Modified dates in every note's footer – is `YYYYMMDD HH:mm`, no dashes, never ISO, because that is what the vault's queries sort and match on. Task and project dates are the one exception, and they cut the other way: `scheduled`, `due`, `start_date`, and `end_date` belong to the TaskNotes plugin and the Gantt views, which expect plain ISO `2026-07-15`. Read those as they are; never reformat them in either direction. Library keys stay Capitalized because the Book Search plugin wrote them that way and `Library.base` reads them that way; rename `Status` to `status` and the book drops off your shelf views. A task's `projects` wikilink places it on the project's Gantt chart; the `#💡` tag in a capture line feeds the `!Idea Incubator` dashboard.

Frontmatter here is an API, not decoration. Claude knows the contract from `CLAUDE.md`, so you never have to remind it; when you edit frontmatter by hand, you are bound by the same rules.

The maturity cycle: 🌱 → 🌿 → 🌳

Every atomic note carries a `maturity` field with exactly one of three values:

VALUE	WHAT IT MEANS
 <code>seedling</code>	The seed of a thought – a couple of lines, almost no links yet
 <code>growing</code>	Context and connections to neighboring notes have appeared
 <code>mature</code>	A polished, self-contained note, well linked

The Card Index base builds its "🌱 Seedlings", "🌳 Mature", and "🔗 By Maturity" views from this field. Claude promotes notes when they earn it – the `link-notes` skill bumps a seedling to 🌿 growing once real connections appear. One rule is hard: maturity never goes down. A demotion needs your explicit say-so.

The aiassist tag: 298 notes you can clear out

The vault ships with 298 demo notes tagged `#aiassist` – sample diary entries, book summaries, atomic notes, contacts. They keep the dashboards and Bases from greeting you with empty screens on day one.

Two things to know. **Review skills warn you:** ask for a weekly review or a sleep trend while demo entries still dominate, and Claude will say so – these numbers come from demo content, not from your life. **They are disposable:** once your own notes fill the views, say "clear out the aiassist notes" and Claude will show you a plan and, after your confirmation, move all 298 out of your way.

TIP

Don't rush the cleanup

Keep the demo notes around for your first week or two. Each one doubles as a live example of correct frontmatter – handy when you create your own notes of the same type.

PART

04

Your First Week

The vault proves itself in a week of light use.
Each day below introduces one skill: what to say,
what happens, and how many minutes it costs.
Nothing here requires discipline – th...

The vault proves itself in a week of light use. Each day below introduces one skill: what to say, what happens, and how many minutes it costs. Nothing here requires discipline – the total workload is under half an hour on the busiest day.

One boundary: this part covers the AI layer only. Obsidian itself – the interface, the plugins, sync, themes – has its own onboarding inside the vault: open `Learning Obsidian/Quick Start/` and walk Day 0 through Day 3, before this week or in parallel with it.

Day 1 – Setup and verify (20 minutes)

Install and connect Claude following Part 2, then run its two verification questions: **"What folders are in my vault?"** and **"Read CLAUDE.md and tell me where a quick idea capture goes."** The first proves Claude can see the files; the second – the right answer is a `#💡` line in `_Inbox/Ideas.md` – proves it has read the contract. If either fails, the fixes are in Part 2. Day 1 isn't about doing anything yet – just confirming the loop is wired, so every skill this week lands files in the right place with the right formatting.

Day 2 – Capture three to five thoughts (5 minutes total)

When a thought worth keeping shows up today – in a conversation with Claude or in your own head – say:

"Capture this: reviewing my own notes beats re-reading the book."

A one-liner becomes a line in `_Inbox/Ideas.md`, in the exact format the vault's dashboards expect: `- #💡 text - 20260713`. A longer thought becomes a small note in `_Inbox/`. Claude picks the destination; if it's a close call, it asks.

Do this three to five times today; each capture costs one sentence and about thirty seconds. In the evening, open `!Idea Incubator.md` in Obsidian – your lines are already there, next to the demo ideas. That's the moment the system stops being theoretical.

Day 3 – Close the day in the diary (10 minutes)

Tonight, before you shut the laptop, say:

"Run my daily review."

Claude creates today's entry in `Periodic Notes/Diary/`, asks you to rate the day on eleven metrics in one compact prompt, then asks one open question: how was your day? Your answer goes in as written. It also surfaces anything you captured today, so nothing rots in the inbox unnoticed.

TIP

The metrics are your personal dataset
Eleven numbers feel like a chore until week three, when the weekly review starts computing averages from them – and month two, when trends appear that you'd never notice day to day. Rate honestly, not precisely. A gut-feel 6 entered every night beats a careful 6.5 entered twice a month.

Day 4 – One thought becomes a card (15 minutes)

Pick the best idea from your Day 2 captures. Say:

"Turn this into an atomic note: reviewing my own notes beats re-reading the book."

Claude drafts the note – one claim, a few sentences in your words, a title that states the point – shows it to you, and on your yes writes it to `Card Index/` with `maturity: 🌱 seedling`. New thoughts always start as seedlings.

Then say: **"Find connections for it."**

The link-notes skill adds one to three links under `## Connections`, each with a clause on why the two notes belong together. Most candidates today will be demo notes tagged `#aiassist` (Part 3 covers them), and Claude flags them as demo when it suggests them. Linking

to samples is fine for practice; your own notes will crowd them out within a month.

Day 5 – Summarize a book (10 minutes)

Open `Library/` and pick any book – the demo shelf has plenty, or add one you've actually read using the Book Search plugin (the vault's Library guide covers it). Say:

"Fill in the summary for Atomic Habits."

Claude fills the empty `# Summary`, `# Quotes`, and `# Key Ideas` sections by interrogating what the book argues – and leaves the frontmatter alone, so the Library views keep working. (Some shipped books title the first section `# Brief Description` instead of `# Summary`; the skill writes under whichever heading the card has.) Read the result and cut anything you disagree with. A book summary you've edited is worth ten you haven't.

Day 6 – Plan a real project (25 minutes)

Not a toy. Pick something you're actually doing this quarter and say:

"Plan a project: launch my portfolio site by the end of August."

This one is a dialogue, not a single command. Claude asks about the outcome, the stages, and what depends on what – then shows the full plan before writing anything. On your yes it creates `Projects/<Name>.md` plus one task note per stage in `Tasks/Tasks/`, each linked back to the project. Open the project note in Obsidian: the Gantt chart has already drawn itself from those tasks. That's the frontmatter contract paying off – you never wired anything.

Day 7 – Review the week (15 minutes)

Say: **"Do my weekly review."**

Claude reads your actual diary entries from the week and drafts the review – wins, loose ends, misses, a focus for next week – into `Periodic Notes/Weekly Reviews/`. With only a few diary entries, it will be

short. Good. This skill never pads: no data means no invented wins, and it says so plainly.

Then look back. Open `!Home.md` and take inventory: a handful of captured ideas, one atomic note with real connections, a book summary, a project with a timeline, a diary trail, and a written review of it all. That's one week, roughly ninety minutes total.

TIP

Stop memorizing phrases

The commands in this part are starting points, not incantations. Skills match intent – "save this," "wrap up my week," and "I want to plan something" all work. When in doubt, just describe what you want; the Skills Cheat Sheet in `Claude/` lists every trigger on one page.

After the week

The honest steady state: capture whenever, ten minutes on the diary at night, fifteen on the review each weekend. Everything else – triage, monthly reviews, week planning, the agents – layers on when you're ready. Part 5 walks through all twelve skills one by one.

PART

05

The 12 Skills, One by One

Every skill in this vault is a folder inside `.claude/skills/`, holding one `SKILL.md` – instructions in plain markdown, nothing compiled, nothing hidden. When you launch Claude Code i...

Every skill in this vault is a folder inside `.claude/skills/`, holding one `SKILL.md` – instructions in plain markdown, nothing compiled, nothing hidden. When you launch Claude Code inside the vault, it discovers all twelve automatically and loads the right one when your request matches its description – no install step. (Claude Desktop does not auto-discover skills – see Part 2.) Because a skill is just markdown, it is yours to change: edit a trigger phrase or tighten a rule, save, and the next session picks it up.

TIP

The one-page version of this chapter – trigger phrases only – lives at `Claude/Skills Cheat Sheet.md`.

1. capture

Say "capture this" or "save this idea." The skill gets a thought out of your head fast, into one of two places. A single phrase becomes a line in `_Inbox/Ideas.md`, in the exact shape the Idea Incubator expects:

```
- #💡 text - YYYYMMDD
```

A developed thought becomes its own note in `_Inbox/`.

You: Capture this – I keep re-reading books instead of reviewing my own notes on them.

Claude: That's a full thought with reasoning, so it gets its own note. Saved to `_Inbox/Review My Notes Instead of Re-Reading Books.md`.

It won't file anything into `Card Index/`, add tags or maturity, or rewrite your phrasing. Sorting comes later – that is inbox-triage's job.

2. atomic-note

Say "make this a note" or "add this to my card index." The skill shapes one thought into an atomic note in `Card Index/`: a title that states a claim, a body of 3–6 sentences in your own words, one to three links to notes that actually exist, and a source. New notes always start as `maturity: 🌱 seedling`.

You: That point about rereading being a trap – make it a note.

Claude: Here's the draft – title: "Rereading Feels Like Learning but Isn't"... One confirmation and I'll save it.

It never writes the file before you approve the draft, never packs two claims into one card, and never sets maturity above seedling – growth happens later, through link-notes.

3. link-notes

Say "link this note" or "find connections." The skill reads the target, searches `Card Index/`, `Library/`, and `Projects/`, and appends one to three lines to `## Connections` – each a wikilink plus one clause on why the two notes belong together. It bumps the footer's Modified stamp and, when a note has earned it, proposes a maturity promotion: 🌱 seedling → 🌿 growing → 🌳 mature. You approve; nothing changes silently.

You: Find connections for "Spaced Repetition Beats Forgetting."

Claude: One new connection holds. The note now has three – promote 🌱 seedling → 🌿 growing?

It never demotes maturity, never forces vague links (zero connections is a valid result), and treats Library and Projects as read-only. When most matches are demo notes tagged `aiassist`, it says so.

4. summarize

Say "summarize this book," "fill in the summary for Atomic Habits," or "TL;DR of this folder." Two modes. For a book card in `Library/`, it fills the empty `# Summary` (some books title it `# Brief Description` – same section), `# Quotes`, and `# Key Ideas` sections by interrogating the author: five questions answered in the author's voice. For any long note or folder, it produces a one-page digest – main idea, key points with wikilinks back to the sources, the strongest quotes, an action.

You: Fill in the summary for Atomic Habits.

Claude: Done – three sections filled, frontmatter untouched. The

quotes are ones I'm confident are verbatim; if you have your own highlights, I'll swap them in.

It never touches the Capitalized Book Search keys (`Status`, `Title`, `Author`...), never invents quotes, and never overwrites your text without showing it first.

5. daily-review

Say "close out my day" or "run my daily review." The skill creates or completes today's entry at `Periodic Notes/Diary/YYYY/MM/YYYYMMDD.md`, asks for the day's eleven metrics in one compact prompt (the exact fields are in Part 3), records your reflection under "How was your day?", then checks `_Inbox/` for today's captures and offers to hand them to inbox-triage.

Claude: Created `Periodic Notes/Diary/2026/07/20260712.md`. Rate your day 0–10.

You: 8 6 7 4 6 5, slept 7.5 quality 7, energy 6 mood 7 stress 4.

It never guesses a number you skipped, never edits past entries, and never pads a quiet day with invented activity – "No captures today" is a complete answer.

6. weekly-review

Say "weekly review" or "wrap up the week." The skill reads the week's seven diary entries and every open task in `Tasks/Tasks/`, then writes `Periodic Notes/Weekly Reviews/YYYY/YYYYMMDD week NN.md` – named after the Sunday – with Wins, Loose Ends, Misses, Discoveries, and a proposed Focus for Next Week. Every bullet traces to something you actually wrote; a section with no data says "No data for this week – fill in by hand."

Claude: Found 6 of 7 diary days – Friday is missing. Focus for Next Week has 3 items – want to change any?

It doesn't coach, doesn't edit your diary, and doesn't change task statuses. Entries tagged `aiassist`? It warns you the review describes the demo author, not you. For multi-week patterns, it offers the weekly-review agent.

7. monthly-review

Say "monthly review" or "close out June." Weekly reviews are the primary source: the skill reads all of them for the month, samples five to eight diary entries, and writes `Periodic Notes/Monthly Reviews/YYYY/YYYYMM.md` – the main event, recurring patterns, how themes evolved week to week, and two or three evidence-based priorities for next month.

Claude: Pattern: "gym" carried over as a loose end in 3 of 4 weeks. Write to `Periodic Notes/Monthly Reviews/2026/202606.md`?

It refuses to manufacture trends – two similar Mondays are not a pattern – and treats every source file as read-only. Zero weekly reviews found? It stops and offers to run weekly-review first. The monthly-review agent is the deeper follow-up.

TIP

The three reviews chain: diary entries feed the weekly review, weekly reviews feed the monthly one. Skip a layer and the next one gets thinner – the skills will tell you so rather than fill the gap.

8. inbox-triage

Say "triage my inbox" or "process my captures." The skill inventories every loose file in `_Inbox/` plus every `#💡` capture line in `Ideas.md`, classifies each item in one line, and shows a numbered routing plan: atomic thoughts to `Card Index/` (via atomic-note), books to `Library/`, multi-step outcomes to `Projects/` plus tasks, conversations to `Periodic Notes/Meetings/`, single actions to `Tasks/Tasks/`, noise to `Archive/`. Then it stops and waits.

Claude: Found 5 items. Here's the plan – confirm and I'll move things... Want to reroute anything?

Nothing moves before you confirm – no exceptions for "obvious" items. Nothing is deleted; junk goes to `Archive/`, only with your ok. It routes without rewriting – improving content is another skill's job.

9. plan-project

Say "plan a project" or "break this goal down." The skill first clarifies three things it won't guess: the outcome (a checkable "done"), the deadline, and your resources. It then proposes five to seven stages with a done-criterion each, flags the riskiest stage to test early, and – only after you confirm – writes `Projects/<Name>.md` plus one task note per stage in `Tasks/Tasks/`, each linked back through `projects: ["[[Name]]"]`. Task dates spread across the project window, so the Gantt inside the project note fills itself.

You: Plan a project: launch my photography portfolio site.

Claude: Two questions first. What does done look like – live with how many galleries? And is there a deadline?

It won't create more than seven tasks up front, and it won't chain tasks with `dependsOn` unless the dependency is real.

10. plan-week

Say "plan my week" or "pick my 6 tasks for tomorrow." The skill crosses two sources: open tasks due or scheduled in the coming seven days (plus anything overdue), and the Focus and Loose Ends from your latest weekly review – your own stated priorities outrank anything it would infer. It proposes up to five priorities, each traced to a task or a review bullet, then writes `Periodic Notes/Plans/YYYYMMDD week NN.md`, named after the Monday. Afterwards it offers the Ivy Lee close: six ranked tasks for tomorrow, in their own note.

Claude: Four priorities – the overdue task first. Want me to turn this into six tasks for tomorrow?

Task notes are read-only input here: no status changes, no date rewrites. Three honest priorities beat five padded ones.

11. obsidian-markdown

This one is a reference, not a workflow. Claude leans on it whenever it writes notes, so everything renders correctly in Obsidian: wikilinks and block references, embeds, callouts, frontmatter

properties, nested tags, footnotes, LaTeX, Mermaid diagrams. You rarely call it by name, but you can.

You: How do I embed just one section of another note?

Claude: `![[Note Name#Heading]]` – the `!` prefix turns any wikilink into an embed.

What it doesn't cover: this vault's own conventions – date formats, maturity values, folder contracts. Those live in `CLAUDE.md`, and the two work together: one knows Obsidian syntax, the other knows your vault.

12. obsidian-bases

Say "edit a base," "add a view," or name any `.base` file. The skill knows the full Bases YAML schema – filters, formulas, views, summaries – plus this vault's specific traps: `maturity` filters match only the exact emoji strings (`"🌱 seedling"`, never bare `"seedling"`), `Library/` keys are Capitalized while movie notes use lowercase ones, and `created` values are `YYYYMMDD HH:mm` strings that `date()` cannot parse. A broken `.base` file fails silently – the view just renders empty – so let the skill validate rather than hand-edit the YAML.

You: Add a "Growing" view to Card Index.base.

Claude: Added a table view filtered on `maturity == "🌱 growing"`, sorted by `created` descending – matching the existing Seedlings view.

It reads the whole `.base` file before adding anything (the view you want may already exist) and never hand-edits the TaskNotes-generated bases – those change through the plugin's settings.

PART

06

Advanced Workflows

This part is for the second month – when the skills feel routine and you want to push further. Everything here works with what's already in the box.

This part is for the second month – when the skills feel routine and you want to push further. Everything here works with what's already in the box.

Three Prompt Patterns Behind the Skills

The skills automate three thinking moves you can also run by hand, with more control.

The contradiction hunt. `link-notes` finds connections for you. The deeper version: hand Claude a fresh Card Index note plus four or five older ones and ask – which ideas build on the old ones, which contradict them and how, which old note deserves an update? Two notes that agree teach you nothing; two that clash mark a spot where your thinking is unfinished.

Interrogating the author. `summarize` writes a summary of a Library book. The deeper version turns it into a conversation. Ask Claude to answer as the author, from the book's content: What was your main idea? What did readers find least obvious? What would you disagree with now? A few answers later you know whether the book earned a permanent note.

Decomposition with a risk probe. `plan-project` builds the project note and its TaskNotes. When a project scares you, decompose by hand first: define "done" in one sentence, cut the work into five to seven stages with a done-criterion each, ask which stage is riskiest – and test that one first. Finish with the MVP question: what can be dropped if time runs short? Then hand the skeleton to `plan-project`.

A Linking Rhythm

A vault grows or rots on two habits, and neither needs more than minutes.

First, cap your inbox debt. Captures pile up in `_Inbox/Ideas.md` all day – one line each, no sorting at capture time. That only works if you clear the debt every two or three days with `inbox-triage`. Miss a week and triage becomes archaeology: Claude routes ten lines in a minute, but forty stale ones need you to remember what half of them meant.

Second, link on a rhythm, not on inspiration. After each writing session, run `link-notes` on what you created. Notes you link within a day stay alive; notes you plan to "link later" join the orphans.

TIP

Claude groups by topic well and guesses your taxonomy badly. Let it cluster and suggest; make the final call – where a note lives, what it's called – yourself. Only what has passed through your own head belongs in the vault.

The Review Chain: Day to Year

The vault reviews on five levels, each feeding the next. Day, week, and month are covered by the `daily-review`, `weekly-review`, and `monthly-review` skills (Part 5). Quarter and year have no dedicated skill, on purpose. Open a note in `Quarterly Reviews/` or `Yearly Reviews/`, ask Claude to read the period's monthlies, and look for what a single month can't show: a loose end that survived three reviews, a metric sliding since spring.

Gardening the Card Index

Promotion through the maturity stages (exact values in Part 3) is earned: a seedling becomes 🌱 growing once it has a developed body and a couple of connections; growing becomes 🌳 mature when other notes link into it and the text stops moving.

Once a month, garden. Open `Bases/Orphan Notes.base`, pick five unlinked notes, and either connect them or admit they were noise. A Card Index where seedlings ripen beats one that only grows wider.

Capture Without Claude

The vault captures on its own. **Alt+Q** opens QuickAdd: the 💡 Idea choice appends a line to `_Inbox/Ideas.md`; Quote, Person, Meeting, and Project choices create full notes from Templater templates, timestamps filled in automatically. No AI involved, works offline, costs two seconds. Claude picks up downstream: `inbox-triage` routes what QuickAdd collected, `link-notes` wires new notes in. One caution:

templates contain live Templater code (`<% %>`) – copy the structure, never the raw code (see Part 3).

Anatomy of an Agent: wise-critic

Skills are procedures; agents are personas with their own context window. Each is one Markdown file in `.claude/agents/`. Here is `wise-critic.md`:

```
---
name: wise-critic
description: "The Critic of the Council of Sages. Stress-tests
  decisions: risks, weak spots, counterarguments, what's been missed.
  Runs after the Visionary to weed out weak options."
tools: Read, Grep, Glob, WebSearch
model: opus
---
```

The body defines the role. The Critic hunts risks and tests assumptions – and it argues from your vault: repeated misses in Weekly Reviews, open tasks, a book note that contradicts the plan. Grounded criticism lands; generic criticism gets dismissed. Note the guardrails: read-only tools, no file edits, and a rule to flag `aiassist` demo notes so sample data never poses as your history.

TIP

Ask "run wise-critic on my plan to switch careers" in Claude Code and the agent gets a fresh context – your main conversation stays clean.

All 8 agents ship with the vault, ready to run; the deep dive into agent workflows – chaining agents, building your own – lives in the Claude + Obsidian course.

PART

07

Migrating an Existing Vault

You probably didn't start from zero – most buyers already have a vault: a year of notes, a structure that grew on its own, plugins tuned just so. This part covers the three ways to...

You probably didn't start from zero – most buyers already have a vault: a year of notes, a structure that grew on its own, plugins tuned just so. This part covers the three ways to bring the AI layer and your existing notes together, and the order of operations that keeps both safe.

Back Up First

Before anything else, close Obsidian. Then:

01 Copy the entire vault folder to a safe place. Not sync – a real copy. Cloud sync happily propagates mistakes; a dated folder like `MyVault-backup-20260715` does not.

02 Copy `.obsidian/` separately. This hidden folder holds your workspace, hotkeys, theme, and every plugin's settings. It is the first casualty of a careless merge, and a standalone copy lets you restore your settings without rolling back your notes.

Open the backup once to confirm it works. Thirty seconds now, hours saved later.

Pick Your Path

YOUR SITUATION	PATH
Young vault, few notes, no structure you're attached to	1 – adopt this vault as your base
Mature vault, structure you like	2 – graft the AI layer onto it
Two full vaults you want in one place	3 – merge

Path 1: Adopt This Vault

The lowest-effort option. Open the Second Brain Vault as your new home, click "Trust author and enable plugins," and move your old notes into a subfolder like `_Inbox/Imported/`. Then let the `inbox-triage` skill route them in batches: "triage twenty notes from `_Inbox/Imported`." Claude proposes a destination for each – Card Index,

Library, Projects, Tasks – and waits for your yes before moving anything.

Everything works on day one because nothing in the vault's contract changed. The cost: you adjust to a folder layout you didn't design.

Path 2: Graft the AI Layer onto Your Vault

Your structure stays; the AI layer adapts to it. Six steps, in this order – the sequence matters.

1. Back up. Done above.

2. Copy three things from this vault into your vault's root: the `.claude/` folder (skills, agents, settings), `CLAUDE.md`, and `.mcp.json`. That's the entire AI layer.

3. Rewrite the folder map in `CLAUDE.md`. This is the step people skip, and it decides whether the skills work. Open `CLAUDE.md`, find the folder map table, and replace every path with your own. Say your vault keeps atomic notes in `Zettel/` and journal entries in `Journal/Daily/`:

```
Before:
| `Card Index/`      | Atomic notes (Zettelkasten)... |
| `Periodic Notes/` | The reflection chain: Diary/... |

After:
| `Zettel/`          | Atomic notes (Zettelkasten)... |
| `Journal/Daily/`  | Daily journal entries           |
```

Delete the rows for folders you don't have. A map that lists a `Library/` you never created teaches Claude to guess, and guessing is where wrong-folder notes come from. The skills navigate by this map; once it describes your vault truthfully, they work without code changes.

4. Copy the templates you need from `Templates/` – only those backing skills you plan to use: `Atomic Note.md` and `Footer.md` for `atomic-note`, `Diary.md` for `daily-review`, `Week Review.md` for `weekly-review`, `Project.md` and `Task.md` for `plan-project`. Templates contain Templater `<% %>` code –

that's normal; skills copy the structure, never the raw code (Part 3).

5. Add frontmatter gradually. Don't batch-convert your whole vault. Skills that create notes bring their own frontmatter; skills that read notes only need the contract on the notes they actually touch. Start with the ten notes you'll work with this week. Copy values exactly: `maturity: 🌱 seedling` with the emoji, `created` dates as `2026071509:30` – no dashes, never ISO. Task and project dates are the exception: `scheduled`, `due`, `start_date`, and `end_date` stay ISO (`2026-07-15`) because TaskNotes and the Gantt views expect them that way. One wrong string silently drops a note from its Base view.

6. Plugins last. The skills write plain Markdown and need no plugins at all. Plugins are the display layer: Bases views, TaskNotes boards, Templater hotkeys. Once the text side works, copy the plugin folders you want from `.obsidian/plugins/` one at a time – keeping your own `app.json` and `appearance.json`, which hold your personal settings.

TIP

Verify before you commit

After step 3, open Claude Code in your vault (Part 2 covers launching) and ask: "List the folders in my vault. Then capture this: test idea." If the capture lands where your rewritten `CLAUDE.md` says captures go, the graft took. If not, the folder map is the first place to look.

Path 3: Merge Two Vaults

For when you want both content sets in one place:

- 01** Decide which vault is the target. If yours is the smaller one, make this vault the target – you're really doing Path 1. If yours is bigger, make it the target – you're doing Path 2 plus a content copy.
- 02** Copy the second vault's notes into the target. Where filenames clash, compare before overwriting and rename the duplicates.

- 03** Copy `.obsidian/plugins/` from this vault to bring the plugin set over, but keep the target's `app.json` and `appearance.json`.
- 04** Run Path 2, steps 3–5, so `CLAUDE.md` matches the merged result.

Migration Questions

Do I have to rename my folders? No. Rename the map, not the folders. `CLAUDE.md` is the single source of truth the skills navigate by.

What about old notes with no frontmatter? Nothing breaks. Claude still reads them, links to them, summarizes them. They just won't appear in Bases views until they carry the contract fields – add those as you touch each note.

How do I clear out the demo content? The demo notes are tagged `#aiassist` – ask Claude to move them all to `Archive/` once your own notes take over (Part 3 has the details).

Something broke mid-migration. Restore the backup from step one and retrace the steps one at a time. Every path here is repeatable.

PART

08

FAQ, Troubleshooting, Privacy

Notes that Claude reads during a conversation are sent to Anthropic's API as part of that conversation – same as pasting them into a chat. Nothing is synced or indexed in the backg...

FAQ and troubleshooting

Claude says it can't see my vault.

For Claude Code: you launched it outside the vault folder – quit, `cd` into the vault root, relaunch. For Claude Desktop: the config path isn't absolute, or the restart wasn't a full quit; Part 2 walks through both fixes. On macOS, also check that the app is allowed to read the folder: System Settings → Privacy & Security → Files and Folders. Test with "list the folders in my vault."

The skills don't trigger when I ask.

Two common causes. You're in Claude Desktop – auto-discovery works only in Claude Code. Or your phrasing drifted too far from the trigger: stay close to the verbs in `Claude/Skills Cheat Sheet.md` – "capture this," "daily review," "triage my inbox" – or name the skill outright: "use the weekly-review skill." If Claude does the task manually instead, the result is usually fine; the skill just guarantees consistent formatting. In Claude Code, also confirm you launched from the folder containing `.claude/skills/`.

A note vanished from a Base view.

Almost always broken frontmatter. Three usual suspects: a `maturity` value without the emoji (`seedling` instead of  `seedling`), a dashed `created` date (`2026-07-12` instead of `20260712 09:30`), or a lowercase key in a Library note (`status` instead of `Status`). Fix the field and the note reappears. If Claude wrote it, tell it so. If the same mistake keeps coming back, make the rule explicit in `CLAUDE.md` and it stops recurring across sessions.

Claude put a note in the wrong place.

Tell it – "this belongs in Card Index as an atomic note" – and it will move the note and fix the frontmatter to match.

My weekly review says I had a great week, but I just unpacked the vault.

You're looking at demo data – the 298 `#aiassist` notes from Part 3. The review skills warn you when their numbers come from demo content. Once your own entries accumulate, archive the demo notes whenever you like.

Privacy: what leaves your machine

Notes that Claude reads during a conversation are sent to Anthropic's API as part of that conversation – same as pasting them into a chat. Nothing is synced or indexed in the background; Claude only reads files when you ask it to do something. By default, Anthropic's API does not use your data to train models. The `claude.ai` web chat has its own setting: Settings → Privacy → "Do not train on my conversations."

Some things still don't belong in the vault at all – or at least not in any folder Claude touches:

- Passwords, API keys, access tokens.
- Documents under NDA. Some NDAs forbid sharing with third-party services, AI included – read yours.
- Other people's personal data without their consent.
- Financial documents that aren't yours to share.

A simple rule before handing Claude a large block of text: would you post it on a server you don't control? If not, keep it in a folder outside the vault – or in a folder you exclude with one line in

```
CLAUDE.md: "never read Private/."
```

Back up before you experiment

An agent that writes files can also overwrite them, so keep a way to roll back. Best option: Git – `git init` in the vault root, `git add -A && git commit -m "backup"` before any bulk operation, `git checkout .` to undo a bad session; the Obsidian Git plugin can commit automatically. No Git? Obsidian Sync keeps version history, Time Machine on macOS restores any file, and a weekly copy to an external drive is crude but foolproof. Losing one note is annoying; losing a month of captures hurts.

Support

Stuck on something this handbook doesn't cover? Reply to your delivery email – the one with your download link. Include what you asked Claude, what happened instead, and which app you used (Claude

Code or Claude Desktop). That's usually enough to diagnose it in one pass.