Second Brain for Engineers

Version 2026 · verified 2026-11-02

Capture is cheap and retrieval is expensive — a plain-Markdown vault with links, few folders, templates, a weekly review and safe agent access.

Contents

Mental model 2

Capture, organise, retrieve 2

Daily and project notes 3

Decision and TIL logs 4

Linking vs tagging 5

Templates 5

Tools: plain Markdown and git 7

Sharing a vault with an AI agent 8

Weekly review and what not to store 9

Further reading 9

Mental model

Capture is cheap and retrieval is expensive. Anyone can dump notes into an app; the system pays off only when you find the right note months later, at the moment you need it. So structure for retrieval — links between notes and a handful of folders — and keep capture friction near zero.

Capture, organise, retrieve

Stage	Do	Avoid
Capture	One inbox, any format, seconds	Filing while capturing
Organise	Weekly: link, title, move	Perfect taxonomies
Retrieve	Search, follow links	Re-reading everything

Tiago Forte's *Building a Second Brain* names four steps — Capture, Organize, Distill, Express (CODE). For engineers the loop that matters is the first two plus retrieval: capture into one inbox, organise in a weekly batch, and write notes so a search hit is understandable on its own. Distil only what you return to.

```
vault/
  inbox/        capture here, no thinking
  daily/        one note per day
  projects/     active work, one folder each
  notes/        evergreen: TILs, how-tos
```

```
  archive/      finished projects
  templates/
```

Daily and project notes

Note	Holds	Lifetime
Daily	Log, scratch, meeting notes	Forever, rarely read
Project	Goal, links, decisions, status	Until shipped, then archive
Evergreen	One idea or how-to, in your words	Revised for years

The daily note is the default capture target: open it, type, link to the project. Obsidian's Daily notes plugin creates a note named after today's date, `YYYY-MM-DD` by default. Project notes collect the links that daily notes point at, so the project note becomes the index for that piece of work.

```
# 2026-11-02

## Log
- 09:40 Paired on [[auth-migration]] token bug
- 14:00 1:1 with Sam — see [[people/sam]]

## Learned
- `git worktree` keeps two branches checked
  out at once -> [[til/git-worktree]]
```

Decision and TIL logs

Log	Records	Pays off when
Decision	Context, options, choice, why	“Why did we do this?”
TIL	One fact, one example, one source	You hit the problem again
Debug journal	Symptom, cause, fix	The bug comes back

Your memory keeps the conclusion and loses the reasoning. A decision log keeps the reasoning, which is what you need when the context changes. TIL notes are short evergreen notes: one fact, a working example, and the link to the primary source.

```
# TIL: git worktree

Check out a second branch in another folder,
sharing one repository.

    git worktree add ../hotfix main

Source: git-scm.com/docs/git-worktree
Related: [[git]]
```

Linking vs tagging

Use	For	Example
Link `[[note]]`	A specific relationship	`[[auth-migration]]`
Tag `#tag`	A status or broad type	`#todo`, `#til`
Nested tag	A small hierarchy	`#inbox/to-read`
Folder	Where the file lives	`projects/`

Links carry meaning — "this bug belongs to that project" — and Obsidian can update them automatically when you rename a note. Tags are for filtering across the vault. Obsidian tags are case-insensitive, cannot contain spaces, and nest with `/`, so a search for `tag:inbox` also matches `#inbox/to-read`.

```
Fixed the retry storm in [[payments-api]];
root cause in [[decisions/2026-10-retries]].

#til #incident
```

> **Gotcha:** Tags grow without limit and drift (`#bug`, `#bugs`, `#Bug`). Keep a short fixed tag list in one note and link instead of tagging for anything specific.

Templates

Variable	Inserts
`{{title}}`	The note's title

Variable	Inserts
`{{date}}`	Today, `YYYY-MM-DD` by default
`{{time}}`	Now, `HH:mm` by default
`{{date:YYYY-MM}}`	Custom Moment.js format

Templates make good structure the path of least resistance: a decision note that starts with the right headings gets filled in. Obsidian's core Templates plugin reads from a folder you set and replaces these variables on insert. Keep templates few and short.

```
---
date: "{{date}}"
status: proposed
---
# Decision: {{title}}

## Context
## Options
## Decision
## Consequences
```

> **Tip:** Quote template variables in properties, as above. Obsidian's help warns the properties editor can overwrite unquoted ones.

Tools: plain Markdown and git

Option	Strength	Cost
Plain files + editor	Zero lock-in	No backlinks
Obsidian	Links, backlinks, search	One more app
Git	History, any device	Merge conflicts on mobile

Choose plain Markdown files on your own disk. Obsidian stores a vault as Markdown files in an ordinary folder, so any editor, `grep` or agent can read it, and you can leave the app without migrating. Version it with git for history.

```
cd ~/vault
git init
printf '%s\n' \
  .obsidian/workspace.json \
  .obsidian/workspaces.json \
  > .gitignore
git add -A && git commit -m "vault: initial"
```

> **Note:** Obsidian rewrites `workspace.json` every time you open a file; its help suggests ignoring it so git stays quiet.

Sharing a vault with an AI agent

Do	Why
Point the agent at specific folders	Less context, fewer leaks
Deny reads on private folders	Notes on people stay private
Keep secrets out of the vault	Agents read what they can reach
Review agent edits as a git diff	Same discipline as code

A Markdown vault is readable context for a coding agent: decision logs and TILs are exactly what it lacks. In Claude Code, `permissions.deny` rules such as `Read(./people/**)` block its file tools from matching paths, but its docs note they do not cover every subprocess — use the sandbox for OS-level enforcement.

```
{
  "permissions": {
    "deny": [
      "Read(./people/**)",
      "Read(./journal/**)",
      "Read(./.env)"
    ]
  }
}
```

> **Warning:** Anything an agent reads can end up in a prompt sent to a model provider. Do not store credentials, health or HR notes in a folder an agent can open.

Weekly review and what not to store

Weekly (30 min)	Never store
Empty the inbox: link, title, file	Passwords, tokens, keys
Archive finished projects	Employer-confidential data on personal disks
Turn one daily-note fact into a TIL	Copies of docs you can link instead

Forte describes reviews at several timescales as part of the cycle; for engineers one fixed weekly slot is enough. Without the review, the inbox becomes the vault and retrieval collapses. Store links to official docs rather than copies — copies go stale silently.

```
## Weekly review — {{date}}
- [ ] Inbox to zero
- [ ] Archive shipped projects
- [ ] One new TIL from daily notes
- [ ] git commit -m "weekly review"
```

Further reading

- Obsidian Help — How Obsidian stores data

- Obsidian Help — Internal links
- Forte Labs — Building a Second Brain
- Forte Labs — The PARA Method
- Zettelkasten.de — Ahrens's note types explained
- Claude Code Docs — Configure permissions