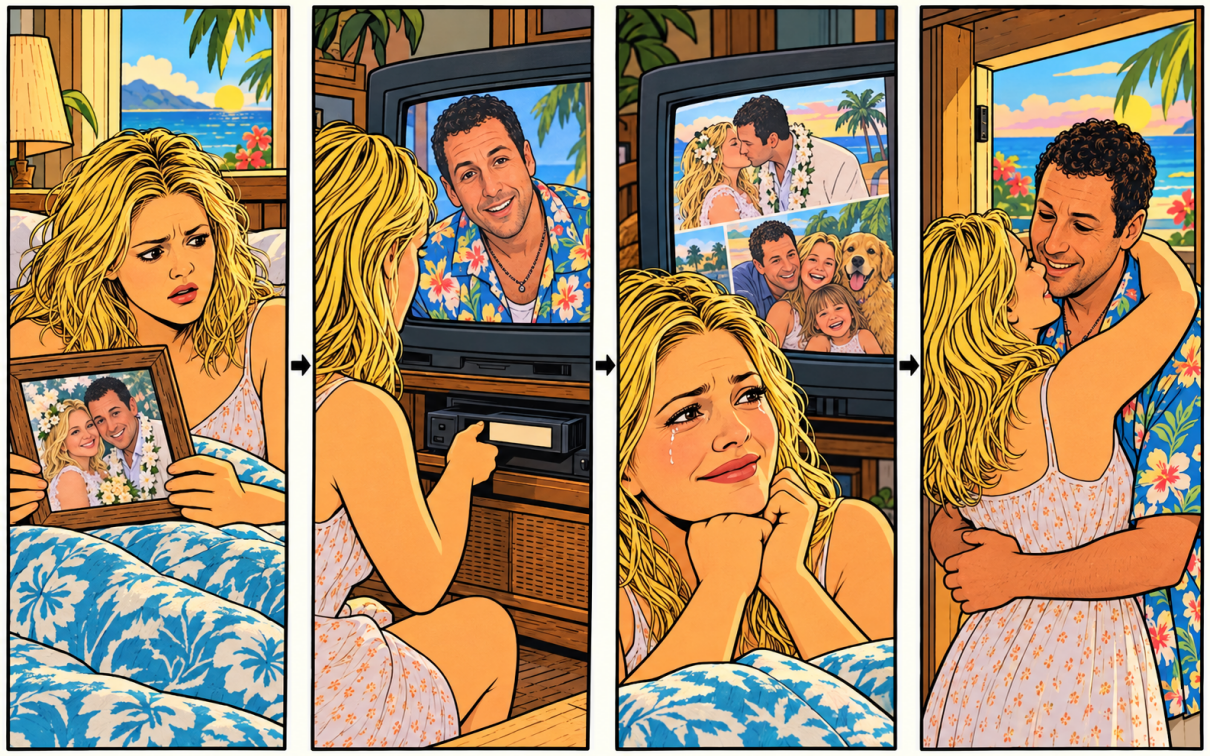
Session 04 AI Workshop

AI × Obsidian

One vault that every agent
can read and write.

Shih-Yang Lin National Dong Hwa University November 11, 2026 HSSB II, D410



Give the next session a way to catch up.

IN THE FILM

A videotape carries her story.

Lucy watches it each morning to learn about her life.

IN YOUR RESEARCH

A vault carries your project's story.

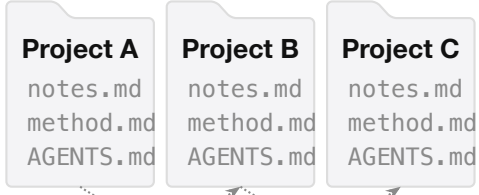
An agent reads it at the start to pick up where you left off.

Keep the decisions, the reasons, and the next step.

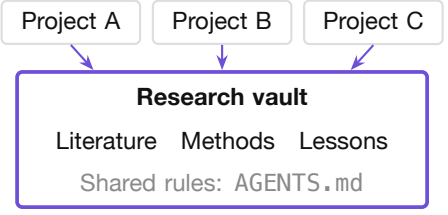
Inspired by *50 First Dates* (2004). Previous slide: AI-generated illustration; adapted scene.

Every project should build on what you already know.

SEPARATE PROJECT FOLDERS



SHARED RESEARCH MEMORY



Shared knowledge. Consistent rules. Project-specific context.

WHERE WE ARE

Session 04 puts the pieces together.

01	02	03	04
Research with AI	AI × Git	AI × Zotero	**Putting It Together**

Git	tracks how your output changes
Zotero	tracks where your claims come from
Obsidian	tracks what you know

TODAY

Three showcases, all on my own research vault.

01	**LLM Wiki**	Feed one paper; watch the agent update ten notes.
02	**Literature Matrix**	Turn notes into a sortable table of papers.
03	**Agent Memory**	One agent writes; another agent remembers.

Five minutes of ideas, then everything runs live.

01

The Idea in Five Minutes

Plain text is the interface.

Pick the notes app your agent can already read.

	NOTION	OBSIDIAN
Where data lives	Vendor cloud	Your disk, as a folder
Data format	Database blocks	Plain Markdown files
Agent access	Cloud API with a token	Any agent that can read a folder

For you alone, both are fine. For you and an agent, files on your disk win.

Criteria from Sessions 01 and 03: open data, an API or CLI, local and scriptable. Check current vendor documentation.

Agents do the note-keeping.

You decide what to keep.

AGENTS

Read sources, write notes, add links, log decisions.

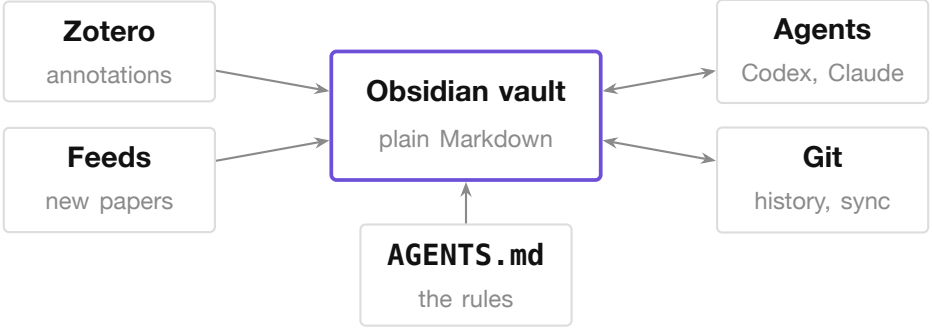
YOU

Review the changes; decide what counts as knowledge.

THE VAULT

Every note and decision, in plain files, with full history.

Files are the only interface between tools.



02

Showcase 1: LLM Wiki

Feed one source. Let the agent maintain the notes. Review the diff.

Raw sources in, a maintained wiki out.

THREE LAYERS

Raw sources never edited

Wiki the LLM writes it

Schema `AGENTS.md`, the rules

THREE OPERATIONS

Ingest one source, 10–15 pages

Query answer with links

Lint contradictions, orphans, gaps

No vector database. An index file and links are enough for hundreds of pages.

Karpathy (2026), "LLM Wiki."

One layer of my vault works like an LLM wiki.

LAYER	KARPATHY	MY VAULT
Raw	`raw/`	`04 Reference Library` 563 Zotero files
Wiki	`wiki/`	`02 Literature`, `03 Permanent`
Schema	`AGENTS.md`	`AGENTS.md` shared by every agent
History	`git`	Obsidian Git auto-sync every 3 minutes
Who writes	the LLM	agents draft; I review the diff

The raw layer comes straight from Session 03.

Project records (Showcase 3) go beyond his pattern.

Ingest one paper.

```
Read AGENTS.md first.

Ingest the annotations for <paper> from 04 Reference Library.
Write a literature note; update the related permanent notes
and the MOC.

Then list every file you changed and why.
```

The rules live in `AGENTS.md`, so the prompt stays short.

The list of changed files is your review checklist.

Keep Obsidian open beside the terminal.

Graph view

New nodes and links appear around the topic.

Backlinks

Old notes now point to the new paper.

`git diff`

Every changed line, ready to keep or discard.

Session 02 pays off here: with git, letting an agent edit your notes is safe.

If AI writes your permanent notes, are you still learning?

Karpathy the LLM writes the wiki

Luhmann you write, in your own words

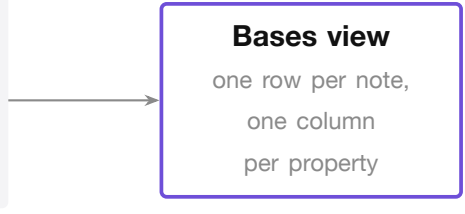
03

Showcase 2: Literature Matrix

The agent fills in properties. Obsidian draws the table.

The agent writes properties. Obsidian draws the table.

```
---
title: CKM (2007)
method: <agent>
data: <agent>
main_finding: <agent>
---
```



Properties are plain YAML at the top of each note. A `.base` file is a saved view over them.

Obsidian Help, “Bases.”

Build the matrix.

```
For every literature note linked from the Business Cycle
Accounting MOC, add these properties: question, method, data,
sample, main_finding.

Use only the note and its Zotero annotations. Write "unclear"
when the source does not say.

Then create BCA Matrix.base with one row per paper.
```

Name the evidence boundary, as in Session 03. “Unclear” is a valid answer; invented details are not.

Five papers, one view.

PAPER	METHOD	DATA	MAIN FINDING
Chari, Kehoe, McGrattan (2007)	live	live	live
Christiano and Davis (2006)	live	live	live
Otsu (2010)	live	live	live
Lama (2011)	live	live	live
Brinca, Costa-Filho, Loria (2024)	live	live	live

The agent judges; the app keeps the record.

NONCODIFIABLE → AGENT

Read each paper's notes

Decide what counts as the method

Summarize the main finding

CODIFIABLE → OBSIDIAN

Store properties exactly

Render the same table every time

Sort, filter, and group instantly

Same split as Session 03.

04

Showcase 3: Agent Memory

Memory that lives in your vault, not inside one agent.

Every agent stores its memory as Markdown.

SYSTEM	RULES	MEMORY
OpenClaw	`AGENTS.md`, `SOUL.md`	`MEMORY.md`, daily files
Claude Code	`CLAUDE.md`	`MEMORY.md` plus topic files
Codex	`AGENTS.md`	rules only
Karpathy	schema file	`index.md`, `log.md`

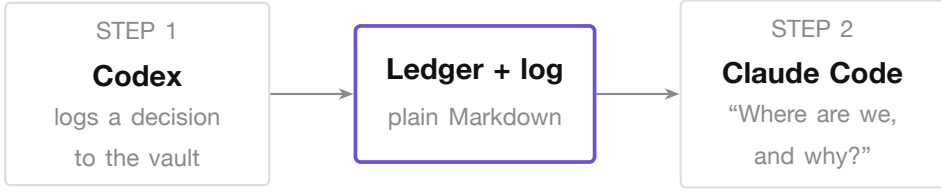
So keep the memory in your own vault.

OpenClaw’s memory, rebuilt in a vault.

OPENCLAW	MY VAULT
Daily memory files	`05 Projects/_logs/` one file per session
`MEMORY.md`	Project ledger, promoted `03 Permanent` notes
Heartbeat	Scheduled jobs e.g., the weekly `08 Feeds` digest
Memory search	Obsidian search, `grep`, Obsidian CLI

OpenClaw workspace layout from its documentation and community guides; file names may change.

One agent writes. Another remembers.



Step 3: `git log` shows both writes, in order. The answer cites the ledger, not a chat history.

What you gain, what you give up.

Gain

- You can read and edit the memory
- Every change has a git history
- Switch agents without losing memory
- No always-on agent with broad access

Give up

- No proactive messages
- No chat through LINE or Telegram
- Agents must be told when to read and write

The missing piece: when to read and write.

DRAFT RULES FOR `AGENTS.MD`

1. Before work: read the project ledger's status section.
2. During work: append each decision to the decision log.
3. Before you stop: write one session log in `05 Projects/_logs/`.

OpenClaw does this automatically; a vault needs it written down. Still being tested.

Records make better delegation possible.

A PATH TO TRY: IMPLEMENTATION COMES LATER

01 Record

Keep context current.

Status, decisions, constraints, and open questions.

02 Brief & delegate

Task + relevant notes.

Specify the output; work locally or send the task remotely.

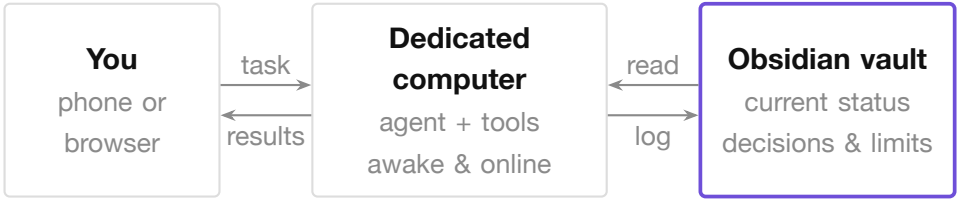
03 Make it routine

Repeat what works.

Regular reviews, one proposed task, and your decision.

The same record supports the next prompt and the next review.

Send the task. Let the vault supply the context.



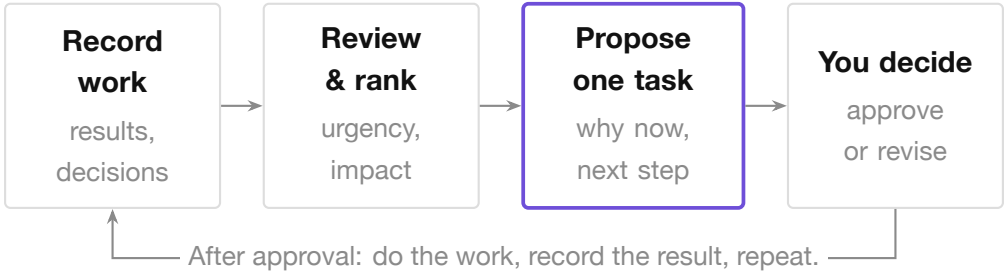
Read Project A's ledger and latest log. Compare the candidate baselines; return one table and the decisions I need to make.

Workflow concept. Remote entry: Codex Remote, Claude Remote Control, or a configured bot.

OpenAI, "Mastering remote engineering work from your phone"; Anthropic, "Remote Control."

Let the agent propose the next task.

CONCEPT: NOT YET IMPLEMENTED



One proposal, with reasons. You set the direction.

“First, reconcile the baseline results.”

ILLUSTRATIVE PROPOSAL: NOT AN AUTOMATED RESULT

Why now?	The ledger says “pending”; a newer log says “complete”.
First step	Compare samples, data vintages, and diagnostics.
Deliverable	One table: the draft’s baseline, the alternatives, and what needs your decision.
Your choice	Approve this check, narrow it, or defer it.

This could prevent the next analysis from using the wrong baseline.

Based on a status mismatch in my vault; project details omitted. Implementation comes later.

05

Putting It Together

Capture, distill, index: the same loop in every system.

The Zettelkasten was a memory protocol all along.

	CAPTURE	DISTILL	INDEX
Zettelkasten	Fleeting notes	Permanent notes	Maps of content
OpenClaw	Daily memory	`MEMORY.md`	—
Karpathy	`log.md`	Wiki pages	`index.md`
My vault	`_logs/`	`03 Permanent`	`00 MOCs`

Agents rediscovered it.

Five rules for composable workflows.

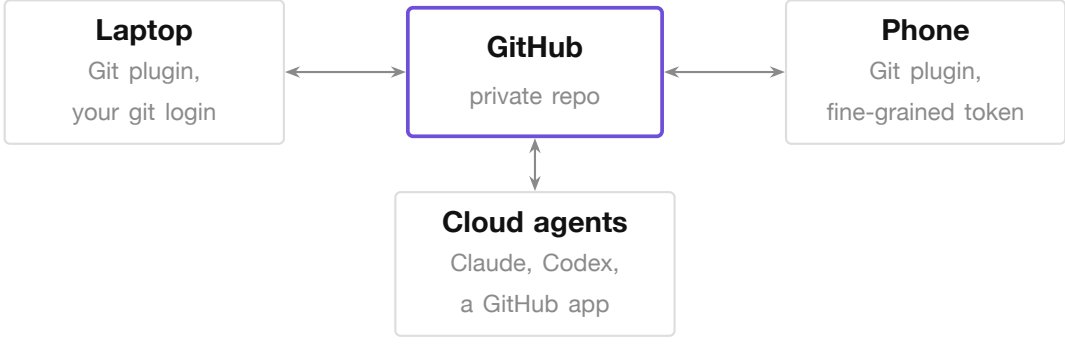
1. **One tool, one job.** Zotero keeps sources; Obsidian keeps notes; git keeps history.
2. **Files are the interface.** Tools talk through Markdown, not through you.
3. **One source of truth for rules.** Other agent files point to `AGENTS.md`.
4. **Gates before writes.** Review the plan or the diff before you keep it.
5. **Version everything.** Any change an agent makes can be undone.

06

Set Up Your Own Vault

A private GitHub repo, synced to your laptop, your phone, and your agents.

One private repo; every device syncs through it.



Git replaces iCloud. Every edit, from any device or agent, becomes a commit.

Two safety rules before you push.

1. IGNORE NOISY AND SECRET FILES

```
.DS_Store
.obsidian/workspace*.json
.obsidian/plugins/*/data.json
```

Plugin settings stay on each device; some hold tokens.

2. PICK ONE SYNC METHOD

Keep the repo out of iCloud and Dropbox

Phone vault: *Store in iCloud* off

Move the old vault out of iCloud

Two sync systems on one folder create conflict copies.

Laptop: clone, open, add one plugin.

1. Create a **private** repo on GitHub.
2. Clone it with GitHub Desktop. (Session 02)
3. Obsidian: *Open folder as vault*, pick the clone.
4. Install and enable the plugin **Git** by Vinzent.
5. Commit the plugin folder, so your phone gets it.

PLUGIN SETTINGS

Auto commit-and-sync interval	3 min
…after stopping file edits	on
Push on commit-and-sync	on
Auto pull interval	3 min
Pull on startup	on
Sync method	merge

Checkpoint: write a note, run *Git: Commit-and-sync*, find it on GitHub.

A token that opens one repo, and nothing else.

GitHub → Settings → Developer settings → Personal access tokens → Fine-grained

Repository access	Only select repositories → your vault
Contents	Read and write
Metadata	Read-only (added automatically)
Expiration	One year

Contents appears only after you pick a repository.

Never use a classic token with full `repo` scope.

An empty vault with one plugin.

1. Create a new vault. Turn *Store in iCloud* off.
2. Install and enable the plugin **Git**.
3. Username: your GitHub account.
4. Token: saved as a secret (any ID, then paste the token).
5. Author name and email: the same as on your laptop.
6. Auto commit, push, and pull: copy the laptop's settings.

"Git is not ready" is normal here: the vault is not a repository until you clone.

Clone into the vault, then restart.

Command palette → *Git: Clone an existing remote repo*

THE PLUGIN ASKS	YOU ANSWER
Remote URL	`https://github.com/<you>/<vault>.git`
Directory	Vault Root
Remote has a `.obsidian` folder?	Yes
Local config conflicts	Delete local config and plugins
Depth of clone	Leave empty (full clone)

Checkpoint: restart Obsidian, edit a note, sync, open it on your laptop.

Last step: tell every agent where notes go.

WITHOUT IT

The agent saves `notes.md` in the repo.

Back to separate project folders.

WITH IT: THREE LAYERS

Global rule	remember to use the skill
Skill `obsidian-notes`	find the vault, write the note
Note policy in the vault	which notes, which folder, ask first?

Change what agents record by editing one note in Obsidian.

Half of the open problem from Showcase 3: when to write.

Install the skills once; every agent gets them.

WHERE NOTES GO

```
git clone \
  https://github.com/errard70101/
  social-science-research-skills
cd social-science-research-skills
python scripts/install.py \
  --skill obsidian-notes --link
```

`--link`: one `git pull` updates every agent.

Windows: `--link` needs Developer Mode.

OBSIDIAN'S OWN SYNTAX

```
npx skills add \
  https://github.com/
  kepano/obsidian-skills
```

Markdown, Bases, Canvas, CLI.

Useful for Showcase 2.

The installer links into `~/.claude/skills`, `~/.agents/skills` (Codex), and `~/.gemini/config/skills` (Antigravity).

SET UP · CONFIGURE

Four settings, done once.

1	**Find the vault**	Nothing to do if the folder is named `obsidian-vault`. Otherwise set `OBSIDIAN_VAULT` to its path.
2	**Note policy**	Save the template as `99 System/Agent Note Policy.md`. It mirrors my folders: rewrite its note types, folders, and templates to fit yours.
3	**Global rule**	Paste the snippet into `CLAUDE.md`, `AGENTS.md`, `GEMINI.md` in each agent's home folder.
4	**Write access**	The vault sits outside your repo; let each agent's sandbox write to it.

Templates and step-by-step setup: `skills/obsidian-notes/references/`.
Optional: Obsidian → Settings → General → Advanced → Command line interface.

Settings 1 and 4, on a Mac and on Windows.

1. VAULT PATH (ONLY IF NEEDED)

```
# Mac, in ~/.zshenv
export OBSIDIAN_VAULT=\
  "$HOME/<vault>"
# Windows PowerShell
setx OBSIDIAN_VAULT \
  "<full path>"
```

4. WRITE ACCESS

```
# Claude Code: ~/.claude/settings.json
"permissions": {
  "additionalDirectories": ["<vault>"]
}
# Codex: ~/.codex/config.toml
[sandbox_workspace_write]
writable_roots = ["<vault>"]
```

Edit these by hand. Claude Code may block an agent from changing its own settings.

Ask for a note; it should land in the vault.

```
We chose 2010 as the base year because the survey changed its
weights in 2009.

Note this down.
```

Expect

The note type, the target path, and a draft to approve. Then a path in the vault and an `obsidian://` link.

Check

`git status` in the repo shows nothing new. The note appears in Obsidian.

Three takeaways.

1. **Keep records others can build on.**
   Turn sources, results, and decisions into records the next session can use.

2. **Give each task the right context.**
   Pair a clear request with relevant records, locally or remotely.

3. **Build routines around review.**
   Let agents propose the next task; you decide what moves forward.

Build your own research vault.

1. Create a private GitHub repo, open it as a vault on your laptop and phone, and turn on Obsidian Git.
2. Write a ten-line `AGENTS.md`: folders, note types, one rule per folder.
3. Ingest three papers from your Zotero library into literature notes.
4. Build one Bases table for a literature you know well.
5. Install `obsidian-notes` and have one agent write a note into your vault.
6. End one research session by asking the agent to write a log to the vault.

Ahrens, Sönke. 2017. *How to Take Smart Notes*. North Charleston, SC: CreateSpace.

Ide, Enrique, and Eduard Talamàs. 2025. “Artificial Intelligence in the Knowledge Economy.” *Journal of Political Economy* 133 (12): 3762–3800.

Karpathy, Andrej. 2026. “LLM Wiki.” GitHub Gist, April 4, 2026. gist.github.com/karpathy/442a6bf555914893e9891c11519de94f.

Obsidian. “Bases.” Obsidian Help. help.obsidian.md/bases.

Vinzent03. “Obsidian Git.” github.com/Vinzent03/obsidian-git.

Ango, Steph (kepano). “Obsidian Skills.” github.com/kepano/obsidian-skills.

Lin, Shih-Yang. “Social Science Research Skills.” github.com/errard70101/social-science-research-skills.

GitHub. “Managing Your Personal Access Tokens.” docs.github.com.