READ MODE · ANSWER FIRST

LLM Wiki Pattern

An interactive, visual explanation of LLM Wiki Pattern, from setup through the complete system.

CHEAT SHEET · 7 ESSENTIAL IDEAS

The whole story in 7 lines

LLM Wiki Pattern becomes easier to reason about when every stage is connected as one system.

1. Here is what to take away from this stage. The problem was never that retrieval is slow or inaccurate.
2. Before we move on, notice what holds this together. It is not the wiki itself. It is the schema.
3. Here is the key thing to notice. When a new source arrives, it does not just get stored in one place.
4. There is a reason most wikis go stale. Keeping them current means updating many interconnected pages every time something changes, and...
5. Did you notice where the agent went first? Not to the raw documents. To the wiki.
6. This is an important habit to build. Lint is not a one-time cleanup. It is a regular pass that keeps the wiki trustworthy as it grows.
7. Here is the conclusion. The wiki gets better every time you use it. Each ingest adds knowledge.

STAGE 1 OF 9

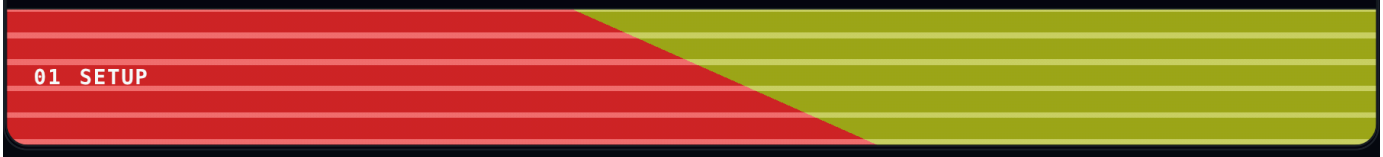
Setup

KEY TERMS

RAG	Look up docs before answering
Wiki	Interlinked pages on topics
Schema	Rules the agent follows
Ingest	Read and weave a source in
Compile	Update, link, and index pages
Lint	Find stale, broken, or missing

ROADMAP

1. Plain RAG Resets
2. Three-Layer Stack
3. Ingest a Source
4. Compile the Wiki
5. Query the Wiki
6. Lint for Gaps
7. Compound the Flywheel



Most people's experience with AI and documents looks something like this: you upload a pile of files, ask a question, and the AI digs through the pile to find an answer. It works, but the AI is rediscovering knowledge from scratch on every question. Ask something subtle that requires combining five different papers, and it has to piece together the fragments every single time. Nothing is built up. Nothing accumulates.

Two terms on the left. RAG stands for Retrieval-Augmented Generation. It is how most AI-plus-documents tools work today: look up relevant chunks, then generate an answer. A wiki is a collection of interlinked pages, each covering one topic and linking to related ones. The core idea of this pattern is that the AI incrementally builds and maintains a wiki for you, instead of re-deriving everything from raw documents on every question.

Two more. A schema is a configuration file you and the AI write together over time. It says how to name pages, what to do when sources disagree, and what workflows to follow. Think of it like the training manual for a new employee. Ingestion is the full process of reading a new document, extracting what matters, and weaving it into the existing wiki. A single source might touch ten to fifteen wiki pages.

The last two. Compiling means running a maintenance pass: revising pages, strengthening cross-links, keeping the index current, flagging contradictions. Linting means checking the wiki for problems: stale facts, orphan pages with no inbound links, claims that newer sources have superseded. Think of it as a health check you run periodically so small cracks never become wrong answers.

Here is the key idea behind these six terms. You never write the wiki yourself. The AI writes and maintains all of it. Your job is to curate sources, ask good questions, and direct the analysis. The AI does the summarizing, cross-referencing, filing, and bookkeeping that makes a knowledge base actually useful over time.

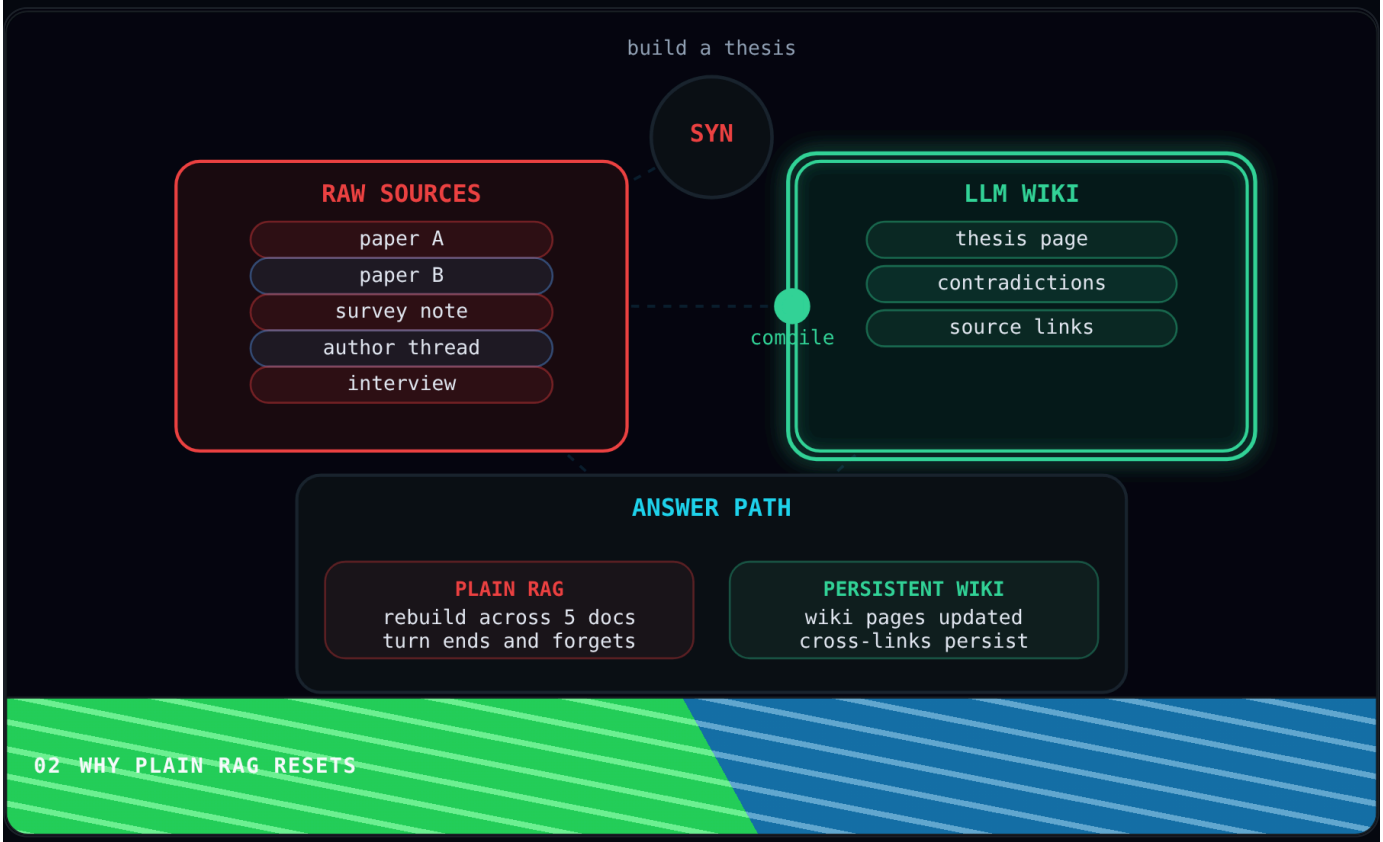
This pattern works for all kinds of projects. Research deep-dives across hundreds of papers. Personal knowledge bases for health, goals, or self-improvement. Reading a book and building a companion wiki

with characters, themes, and plot threads. Business teams fed by Slack threads, meeting transcripts, and project documents. Anything where you accumulate knowledge over time and want it organized, not scattered.

The roadmap on the right shows our seven stops. We will start by seeing why plain RAG falls short. Then we will build the three-layer architecture, follow a document through ingestion, see a maintenance pass, run a query, learn about linting, and watch the whole flywheel spin. Let us start with the most basic question: what goes wrong when every answer is built from scratch?

STAGE 2 OF 9

Why Plain RAG Resets



A question arrives at the top and the system goes straight to the RAW SOURCES panel to hunt for relevant pieces. This is how NotebookLM, ChatGPT file uploads, and most RAG tools work. You ask, it searches, it answers. Seems fine.

The system stitches together an answer from raw fragments and hands it over. Then it forgets everything it just figured out. Next question? Same pile, same digging, same cost. Every cross-document connection, every synthesis, rebuilt from scratch. Nothing from the last answer carries forward.

Look at the ANSWER PATH panel at the bottom. The PLAIN RAG column on the left shows the cost. Switch the Question control to Fact. Even a simple lookup scans raw docs. Switch to Synthesis and the cost jumps because the system has to piece together multiple sources from scratch every time. Increase the Docs stepper to 6 and watch the source panel fill up.

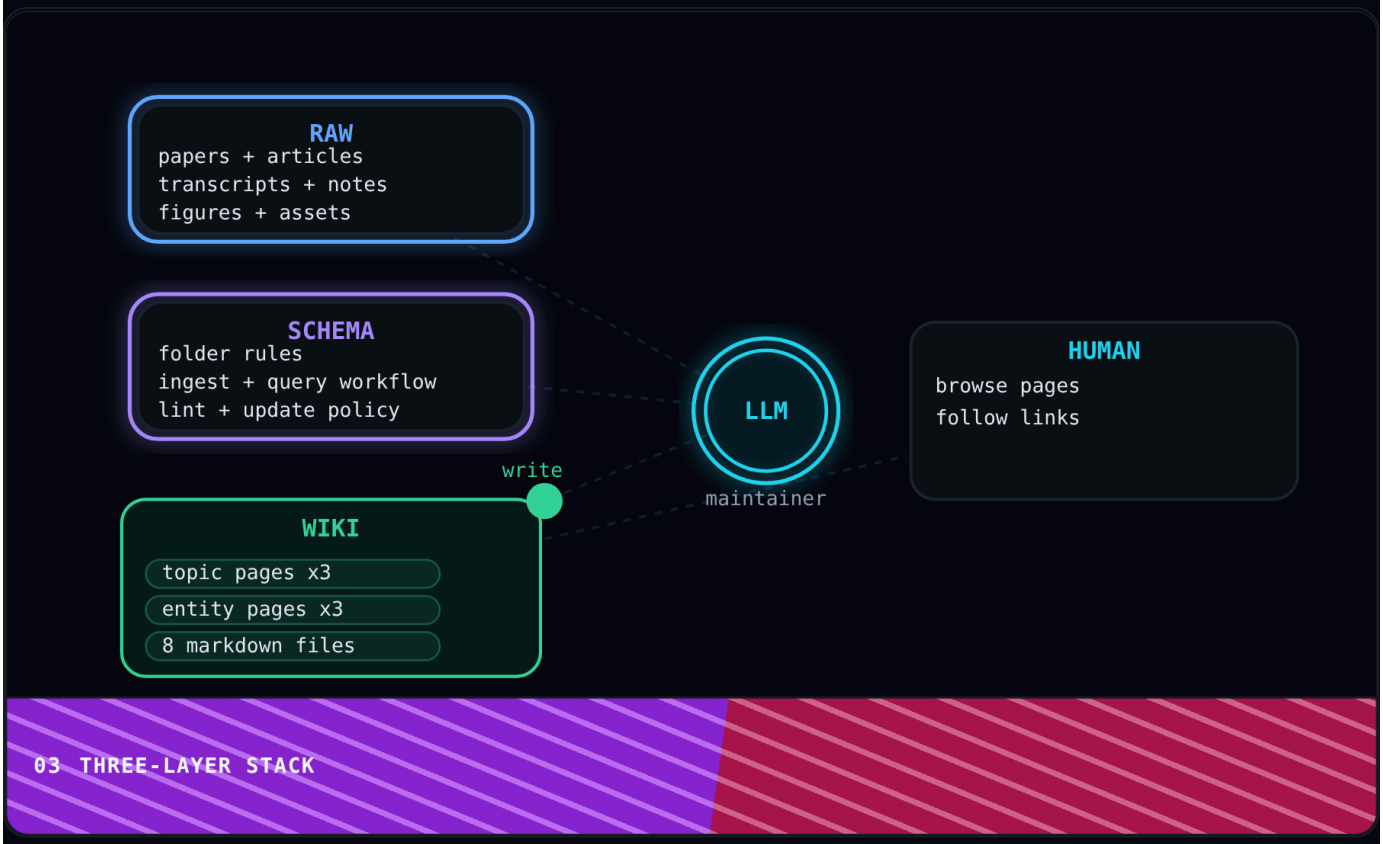
Now look at the PERSISTENT WIKI column on the right. When a wiki exists, the hard work of reading, connecting, and synthesizing documents has already been done in earlier sessions. The cross-references are already built. The contradictions have already been flagged. The synthesis already reflects everything you have read.

Toggle the Wiki control off. The LLM WIKI panel goes dark and the answer path falls back to raw retrieval. Toggle it on and the compiled pages return. That single toggle is the difference between re-deriving knowledge from scratch every time and reading from a persistent, compounding artifact.

The problem with plain RAG is not speed. It is that nothing accumulates. The wiki is the fix: a persistent layer where knowledge is compiled once and kept current, not re-derived on every question. Now let us see what that wiki actually looks like inside.

STAGE 3 OF 9

Three-Layer Stack



Three boxes stacked on the left, each with one job. The RAW layer at the bottom holds your curated source documents: articles, papers, transcripts, images. These are immutable. The AI reads from here but never modifies anything. Your originals stay clean.

Switch the Sources control between Research, Personal, and Team. The contents change but the architecture stays identical. Academic papers, personal journals, Slack threads, meeting transcripts, customer calls. The three-layer pattern works for all of them.

The middle layer is the SCHEMA. Remember in the Setup when we called it a training manual? This is where it lives. In practice it is a file like CLAUDE.md or AGENTS.md that tells the AI how to name pages, what to do on ingest, how to handle contradictions. You and the AI co-evolve this file over time as you figure out what works for your domain. Switch Schema to Light and watch the rule set shrink.

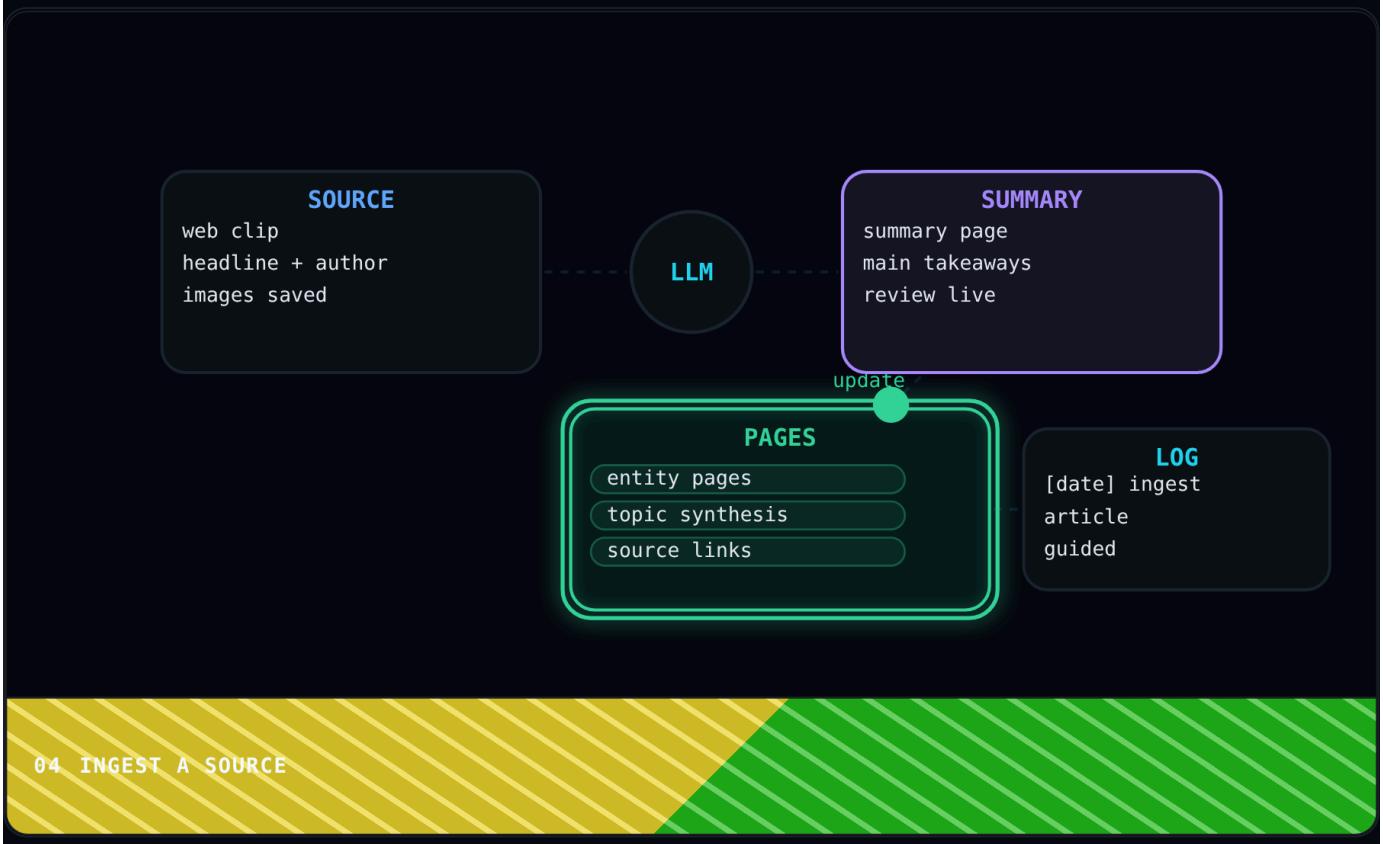
Watch the LLM circle on the right. It reads rules from the schema and raw material from the sources, then writes to the wiki. The schema is what turns the AI from a generic chatbot into a disciplined wiki maintainer. Without it, every session would produce different conventions.

The top layer is the WIKI. This is the compiled product: markdown pages, cross-links, summaries, entity pages, concept pages. The AI owns this layer entirely. It creates pages, updates them, maintains cross-references, and keeps everything consistent. You read it; the AI writes it. Use the Pages stepper to watch the wiki grow.

Three layers, three owners. Raw sources are yours and stay untouched. The schema is co-owned and evolves with your needs. The wiki is the AI's domain. The whole thing is just a git repo of markdown files, so you get version history, branching, and collaboration for free. Now let us see what happens when a new document arrives.

STAGE 4 OF 9

Ingest a Source



A new document has landed in the SOURCE panel. Maybe you clipped an article from the web using Obsidian Web Clipper, or dropped a paper PDF into the raw directory. In the plain RAG world we saw in Stage 1, this file would just sit in the pile. Here, something very different happens.

The AI reads the full document. Not just keywords. It extracts key claims, important details, metadata, and discusses the takeaways with you. Switch the Source control to Paper and notice the extracted information changes shape. Switch to Transcript and it changes again. Each source type carries different kinds of value.

Next the AI writes a SUMMARY page in the wiki. This is not a filename sitting in a folder. It is a real wiki page that captures what the source says and why it matters, with links to the related concept and entity pages. You read the summary in Obsidian, check it, and guide the AI on what to emphasize.

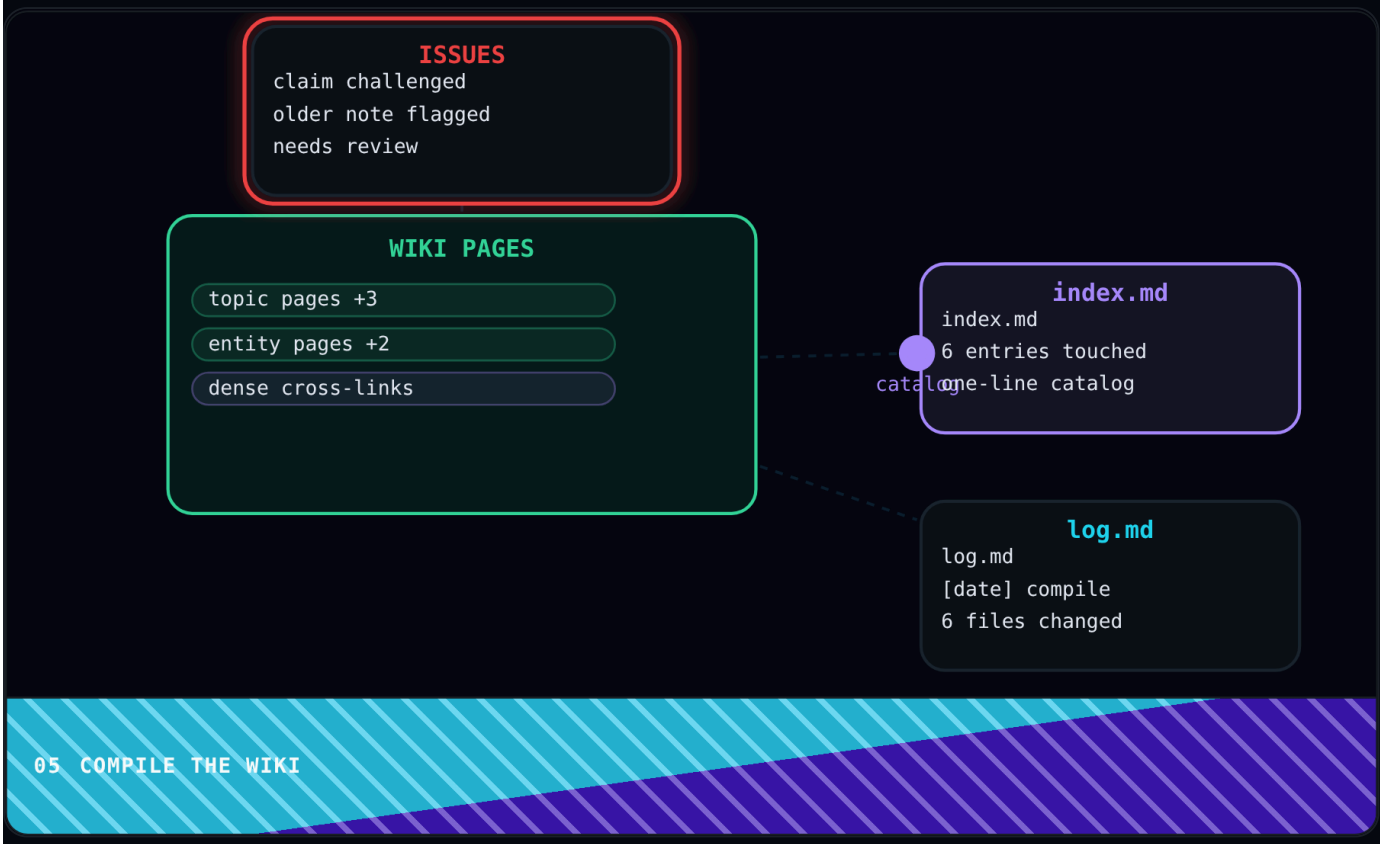
Now the most important part. The AI walks through every existing wiki page that overlaps with this source and updates each one. A new finding about a concept lands on that concept's page. A new detail about a person lands on that person's page. A single source can touch ten to fifteen wiki pages. Switch Supervision to Batch to see how multiple sources can be processed in one go with less hand-holding.

The ingest gets recorded in the LOG with a date: what was added, what pages were touched, what changed. Toggle the Images control off and visual content stops being downloaded separately. Toggle it on and images get saved locally so the AI can reference them directly instead of relying on URLs that might break.

The document has been read, summarized, and woven into the existing wiki across multiple pages. The knowledge is compiled once, not waiting to be re-derived on every future question. But those updated pages now need a consistency check. Let us see what a maintenance pass looks like.

STAGE 5 OF 9

Compile the Wiki



The WIKI PAGES panel is the heart of the system. After ingestion touches multiple pages, the AI re-reads them to check consistency. Remember in Stage 1 how plain RAG rebuilt everything from scratch every time? Here the cross-references are permanent. The AI just needs to make sure they are still accurate.

Every page that overlaps with the new material gets revised. Use the Touches stepper to increase the count and watch more page rows light up. A single new article can ripple through half the wiki, updating topic summaries, entity pages, and syntheses. This is the tedious bookkeeping that kills hand-maintained wikis. The AI does it in one pass.

Cross-links between pages get strengthened. Switch Links to Few and the wiring thins out. Switch to Dense and the connections multiply. These links are what make the wiki navigable in Obsidian's graph view. You can see which pages are hubs, which are orphans, and how ideas connect at a glance.

Here is where the wiki earns trust. When new evidence contradicts something already in the wiki, the AI does not silently overwrite the old claim. It flags the conflict in the ISSUES panel so you can see the disagreement. Toggle Conflict off and the panel empties. Toggle it on and you can see honest record-keeping in action.

The index.md file gets refreshed. This file is content-oriented: a catalog of every page with a one-line summary, organized by category. When the AI answers a question later, it reads this index first to find relevant pages, then drills in. This works surprisingly well even at around a hundred sources without needing any embedding-based RAG infrastructure.

A dated entry in log.md closes the pass. The log is chronological: what happened and when. Ingests, queries, lint passes, all timestamped. You can grep through it to see recent activity. Now the wiki is maintained, the index is current, and contradictions are surfaced. Let us actually use it to answer a question.

STAGE 6 OF 9

Query the Wiki

A question arrives at the ASK circle. In Stage 1 we watched plain RAG dig through raw documents every time. This time the path is completely different. Watch where the AI goes first.

It goes to index.md, the catalog we just refreshed during compilation. The index tells the AI which pages are relevant, so it opens only the ones it needs. Use the Pages stepper to raise the count and watch more page rows appear. Switch Question to Thesis and the AI pulls in more pages to build a deeper argument.

Here is the payoff for all the earlier work. Those pages were written during ingestion and kept current during compilation. The synthesis is already there. The cross-references are already built. The contradictions have already been flagged. The AI reads what exists instead of rebuilding it from raw fragments. All that maintenance was the investment. This is the return.

The ANSWER panel assembles a response with citations pointing back to specific wiki pages. You can click through to the cited pages in Obsidian and read the full context. No vague references to documents you would have to hunt down.

Answers do not have to be plain text. Switch Output to Table and the answer becomes a comparison grid. Switch to Slides and it becomes a Marp slide deck you can view right in Obsidian. The same compiled wiki pages power all three formats. You get answers as markdown files, charts, presentations, whatever fits the question.

Here is the crucial insight: good answers should not disappear into chat history. If an answer is worth keeping, it can be filed back into the wiki as a new page. A comparison you asked for, an analysis, a connection you discovered. Your explorations compound in the knowledge base just like ingested sources do. But a growing wiki can also develop problems. Let us see how to catch those.

STAGE 7 OF 9

Lint for Gaps

The GRAPH SLICE panel shows a piece of your wiki as a network of nodes and connections. This is what Obsidian's graph view looks like in practice: you can see which pages are hubs, which are orphans, and where the connections are thin. Remember the schema from Stage 2? It keeps conventions consistent, but it cannot prevent every kind of drift over time.

The LINT circle walks through every page systematically. Switch the Check control to Orphans and it hunts for pages with no inbound links, pages that exist but nothing points to. Switch to Stale and it looks for claims that newer sources have superseded. Switch to Conflicts and it finds pages that contradict each other.

What it finds becomes a concrete list in the ISSUES panel. Not vague hunches, but specific problems you can act on. "This page claims X, but that page says the opposite." Or "this concept appears on five pages but has no page of its own." Or "this claim has not been revisited since three new sources were added."

Use the Graph stepper to add more nodes and watch the network fill in. With more pages visible you can see where connections are dense and where they are sparse. The AI is good at noticing structural gaps that are hard to spot by reading pages one at a time.

The AI can also suggest what to read or search for next. Toggle Search on and the NEXT SEARCH panel lights up with specific recommendations. A thin spot in the wiki becomes a concrete reading list. The AI is surprisingly good at suggesting further questions to investigate and new sources to look for.

Linting keeps the wiki healthy as it scales. At a hundred sources and hundreds of pages, you cannot manually check everything. But the AI can run a health check in one pass and surface the problems worth your attention. Now we have all four operations: ingest, compile, query, and lint. Let us see what happens when they run together.

STAGE 8 OF 9

Compound the Flywheel

Every session starts at the YOU circle. You show up with a direction: a new article to ingest, a question to explore, or a hunch that some pages need a checkup. The QUEUE below holds whatever is waiting. Use the Backlog stepper to add items and watch the queue grow. Your job is to curate sources, direct the analysis, and ask good questions.

The OBSIDIAN panel is your window into the wiki. You browse pages, follow links, check the graph view, read updated summaries. Obsidian is the IDE. The AI is the programmer. The wiki is the codebase. The AI edits markdown files based on your conversation, and you see the changes land in real time. Switch Workflow to Team to see how multiple people can feed the same wiki.

The WIKI panel is where all four operations converge. Ingestion adds pages. Compilation revises and links them. Querying reads from them. Linting checks them for problems. All four write back to the same shared structure of markdown files.

Notice the dashed line from ANSWERS back to WIKI. This is the compounding trick we introduced in the query stage. Toggle Capture off and answers disappear into chat history. Toggle it on and useful answers become permanent wiki pages. A comparison you ran, an analysis you asked for, a connection you spotted. Your explorations accumulate in the knowledge base just like ingested sources do.

Every time you come back, the wiki is richer than when you left. The ten-source wiki becomes a fifty-source wiki, then a hundred-source wiki with hundreds of pages and hundreds of thousands of words. Questions that once required piecing together five raw papers now get answered from a single compiled page.

The reason most hand-maintained wikis die is that the maintenance burden grows faster than the value. Updating cross-references, keeping summaries current, noting contradictions, maintaining consistency across dozens of pages. Humans give up. The AI does not. It never gets bored, never forgets to update a cross-reference, and can touch fifteen files in one pass. You bring the curiosity and judgment. The AI brings the patience. Let us step back and see the whole picture.

Recap

We started with the problem. Plain RAG forgets everything between questions. It digs through raw documents, stitches together an answer, and throws away all the work. NotebookLM, ChatGPT uploads, most RAG tools. Same pattern, same limitation. Nothing accumulates.

The fix was a three-layer stack. Raw sources at the bottom stay immutable, your source of truth. The schema in the middle turns the AI from a generic chatbot into a disciplined wiki maintainer. The wiki at the top is a persistent, compounding artifact of markdown files that the AI writes and you browse.

Ingestion is how new knowledge enters. The AI reads a source, writes a summary page, and updates every related page across the wiki. A single source can touch ten to fifteen pages. The knowledge is compiled once and kept current, not re-derived on every future question.

Compilation is the maintenance pass. Pages get revised, cross-linked, and indexed. Contradictions get flagged, not buried. The index.md catalog gets refreshed so the AI knows what exists. This is the tedious bookkeeping that kills hand-maintained wikis, and the AI does it every single time.

Querying is where the investment pays off. The AI reads the wiki first, not the raw pile. The synthesis is already there, the links are already built. Answers come back as markdown pages, comparison tables, slide decks, whatever fits the question. And good answers get filed back into the wiki so your explorations compound.

Linting catches the cracks before they cause wrong answers. Contradictions between pages, stale claims, orphan pages, concepts mentioned everywhere but lacking their own page. The AI turns vague unease into a concrete checklist and even suggests what to search for next.

The flywheel connects all four operations. Each ingest adds knowledge. Each query can grow the wiki with a new page. Each lint pass removes rot. The wiki compounds because the AI does the maintenance that humans would skip. The ten-source wiki becomes a hundred-source wiki, and the answer path gets cheaper every time.

Here is the one idea worth remembering. The LLM Wiki Pattern works because it pairs human judgment with machine patience. You choose what matters, curate sources, ask the right questions, and direct the analysis. The AI handles the summarizing, cross-referencing, filing, and bookkeeping that nobody wants to do by hand. Together, the knowledge base gets smarter every single time you use it.