Obsidian + Local LLM: Build Your Private AI Second Brain

February 25, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Install Ollama, pull a chat model (Qwen 2.5 14B or 7B) and an embedding model (nomic-embed-text), then connect Obsidian using one of three paths. Smart Connections is the easiest — install the plugin, point it at Ollama for chat, and its built-in embedding model indexes your vault automatically. Open WebUI gives you a separate chat interface with RAG over uploaded vault files. AnythingLLM offers workspace-based RAG with a desktop app. All three keep your notes fully local. Smart Connections is the right starting point for most people.

Related: Local RAG Guide · Embedding Models for RAG · Obsidian + Copilot Setup · VRAM Calculator

Your notes are the most personal data you own. Research, journals, project plans, half-finished ideas — all of it sitting in an Obsidian vault as plain markdown files on your disk. Cloud AI tools want you to upload that vault to their servers. That should bother you.

The alternative: run a local LLM on your own machine and connect it to Obsidian. Your notes stay on your hardware, no API keys leak your data, and the AI layer works offline. You get semantic search, note Q&A, summarization, and connection discovery across your entire vault without sending a single word to anyone's cloud.

There are three practical ways to set this up, each with different tradeoffs. This guide covers all three, from easiest to most involved.

The three paths

Path	Setup time	Best for	Vault sync	Runs inside Obsidian?
Smart Connections + Ollama	10 minutes	Semantic note linking, in-Obsidian chat	Automatic (real-time)	Yes
Open WebUI + Ollama	20-30 minutes	Separate chat UI, multi-model switching, teams	Manual upload or scripted	No (browser UI)

Path	Setup time	Best for	Vault sync	Runs inside Obsidian?
AnythingLLM + Ollama	15-20 minutes	Desktop app, workspace RAG, agent features	Manual or third-party script	No (separate app)

All three require Ollama running locally. Start there.

Prerequisites: Ollama + models

If you already have Ollama running, skip to the path that interests you. Otherwise:

Install Ollama:

```
# Linux
curl -fsSL https://ollama.com/install.sh | sh

# macOS — download from ollama.com or:
brew install --cask ollama

# Windows — download installer from ollama.com
```

Pull your models:

```
# Chat model (pick one based on your VRAM)
ollama pull qwen2.5:7b          # 8GB VRAM — ~5GB download
ollama pull qwen2.5:14b         # 12-16GB VRAM — ~9GB download

# Embedding model (required for RAG)
ollama pull nomic-embed-text    # ~274MB, fast, 8K token context
```

Verify the server is running:

```
curl http://localhost:11434
# Should return: Ollama is running
```

If it's not running, start it with `ollama serve`. On macOS, launching the Ollama app starts the server automatically.

CORS fix (needed for Obsidian plugins):

Obsidian plugins connect to Ollama from inside the app, which triggers browser-style CORS restrictions. Allow the Obsidian origin:

```
# Linux/Windows
OLLAMA_ORIGINS=app://obsidian.md* ollama serve

# macOS (persistent)
launchctl setenv OLLAMA_ORIGINS "app://obsidian.md*"
# Then restart the Ollama app
```

Path 1: Smart Connections + Ollama (easiest)

Smart Connections is an Obsidian plugin that embeds every note in your vault as a vector and uses those embeddings to show semantically related notes in a sidebar. It also has Smart Chat, a RAG-powered chat feature that answers questions from your notes.

Why start here: it runs entirely inside Obsidian, indexes your vault automatically, and the free tier gives you working semantic search with zero configuration.

Install

1. Open Obsidian > Settings > Community Plugins.
2. Click Browse, search for "Smart Connections" by Brian Petro.
3. Install and enable.

The plugin starts indexing your vault immediately using its built-in embedding model (multilingual-e5-small, running via Transformers.js in the browser). No Ollama needed for this part. Embeddings are stored locally in a `.smart-env/` folder inside your vault.

Indexing time depends on vault size:

- 100 notes: under a minute
- 500 notes: 3-5 minutes
- 2,000+ notes: 15-30 minutes (first run only)

Configure Ollama for chat

The built-in embedding handles search and connections. For Smart Chat (the Q&A feature), point it at your local Ollama model:

1. Settings > Smart Connections > scroll to the Chat section.
2. Set **Model Platform** to "Custom Local (OpenAI format)."
3. Configure:
    - **Model Name:** `qwen2.5:14b` (or whichever you pulled)
    - **Protocol:** `http`
    - **Hostname:** `localhost`
    - **Port:** `11434`
    - **Path:** `/api/chat`

Use it

Open the Smart View from the left ribbon or command palette (Ctrl/Cmd+P > "Open: Connections view"). While you work on any note, the sidebar shows semantically related notes across your vault, ranked by similarity score. This catches connections that keyword search misses entirely. Two notes about "model quantization" and "reducing inference memory" will surface as related even if they share no common words.

For Q&A, open Smart Chat from the command palette. Ask questions like:

- "What have I written about RAG chunking strategies?"
- "Summarize my notes on transformer architectures"
- "Find contradictions between my notes on GPU benchmarks"

Smart Chat retrieves the most relevant note chunks via embedding similarity, feeds them to your Ollama model as context, and generates a grounded response.

What to know

- The free tier uses the built-in browser-based embedding model. Ollama-based embeddings (using nomic-embed-text) are available in the paid tier (~$10-15/month via smartconnections.app). For most vaults, the built-in embedder works fine.
- Connections update automatically as you edit notes. No manual re-indexing.
- Settings > Smart Connections > Show environment settings lets you choose between "Sources" (whole-note embeddings) and "Blocks" (paragraph-level). Blocks give more granular results but create more vectors.

Path 2: Open WebUI RAG (most flexible chat interface)

Open WebUI is a self-hosted chat interface that looks and feels like ChatGPT but connects to local Ollama models. Its Knowledge Base feature lets you upload documents and chat with them using RAG. You upload your Obsidian vault files, and Open WebUI chunks, embeds, and stores them for retrieval.

Why choose this: if you want a polished chat UI, the ability to switch between multiple models mid-conversation, or you're already running Open WebUI for other local AI tasks. The tradeoff is that your vault files need to be uploaded separately — Open WebUI doesn't live inside Obsidian.

Install Open WebUI

```
docker run -d -p 3000:8080 \
  --add-host=host.docker.internal:host-gateway \
  -v open-webui:/app/backend/data \
  --name open-webui \
  ghcr.io/open-webui/open-webui:main
```

Access it at `http://localhost:3000`. Create an admin account on first launch. It auto-detects your local Ollama instance.

Configure the embedding engine

By default, Open WebUI uses `sentence-transformers/all-MiniLM-L6-v2` for embeddings, which runs on CPU inside the container. For better quality with longer notes, switch to Ollama:

1. Admin Panel > Settings > Documents.
2. Set **Embedding Engine** to "Ollama."
3. Set **Embedding Model** to `nomic-embed-text`.
4. Ollama should auto-connect at `http://host.docker.internal:11434` from inside Docker.

Optional tuning in the same panel:

Setting	Recommended value	Why
Chunk Size	1500 characters	Balances context vs precision for markdown notes
Chunk Overlap	200 characters	Preserves context across chunk boundaries

Setting	Recommended value	Why
Top K	5-8	Number of chunks retrieved per query
Hybrid Search	On	Combines semantic + keyword (BM25) matching

Upload your vault

1. Go to Workspace > Knowledge > click **+** to create a new Knowledge Base.
2. Name it (e.g., “Obsidian Vault”), add a description.
3. Upload your `.md` files. Open WebUI supports directory upload — drag your vault folder and it recursively uploads markdown files (up to 100 files per batch by default, configurable via `FOLDER_MAX_FILE_COUNT`).
4. Wait for processing. Embedding happens in the background.

To use the knowledge base in chat, type `#` in the chat input and select your knowledge base from the dropdown. Open WebUI retrieves relevant chunks and injects them as context for the model.

The sync problem (and workarounds)

Open WebUI does not watch your vault folder for changes. When you add or edit notes in Obsidian, they don’t automatically appear in your knowledge base. Options:

- **Manual re-upload:** Simple but tedious for active vaults.
- **API script via cron:** Write a Python script that calls the Open WebUI API to upload new/modified files. Run it every 15-30 minutes via cron. The community has several examples on GitHub.
- **MCP bridge:** The obsidian-vault-mcp tool exposes vault files via MCP protocol, which Open WebUI can consume.

For vaults that don’t change frequently (reference material, archived research), manual upload works fine. For daily notes and active projects, the sync gap is a real friction point.

Pair a model with your knowledge base

1. Workspace > Models > click **+** to create a custom model.
2. Set a name (“Vault Assistant”), pick your base Ollama model, and attach your Knowledge Base.
3. Now selecting this model in chat automatically includes vault context without typing `#` each time.

Path 3: AnythingLLM (best desktop RAG app)

AnythingLLM is a standalone desktop application (also available as Docker) that wraps Ollama with a workspace-based RAG interface. Each workspace is a collection of documents that the model can search and reference. You create a workspace for your Obsidian vault and chat with it.

Why choose this: if you want a dedicated desktop app for vault RAG rather than a browser tab or an Obsidian plugin. AnythingLLM has workspace isolation (different document collections for different projects), agent capabilities, and a clean interface. The tradeoff: like Open WebUI, it's a separate application from Obsidian.

Install

Desktop (easiest): Download from anythingllm.com/desktop. Available for Windows, macOS, and Linux. Install like any app.

Docker:

```
docker run -d -p 3001:3001 \
  --add-host=host.docker.internal:host-gateway \
  -v anythingllm:/app/server/storage \
  --name anythingllm \
  mintplexlabs/anythingllm
```

Connect to Ollama

On first launch, AnythingLLM asks you to configure an LLM provider:

1. Select **Ollama** as the LLM provider.
2. Base URL: `http://localhost:11434` (or `http://host.docker.internal:11434` for Docker).
3. Select your chat model from the dropdown (e.g., `qwen2.5:14b`).

Configure embeddings

AnythingLLM ships with a built-in embedding model (`all-MiniLM-L6-v2`, ~25MB, CPU-only). It works but has a 512-token context limit, which means long notes get chopped into tiny chunks. For better results with markdown files:

1. Settings > Embedder > select **Ollama**.
2. Set the base URL to `http://localhost:11434`.
3. Select `nomic-embed-text` from the dropdown.

One important restriction: the embedding model is system-wide, not per-workspace. Changing it after you've already embedded documents means deleting and re-embedding everything. Pick your embedder first, then start uploading.

Create a workspace and upload your vault

1. Click **+** in the left sidebar to create a new workspace.
2. Name it (e.g., "Obsidian Vault").
3. Set chat mode to **Query** — this restricts responses to information found in your documents, which is what you want for vault Q&A. The alternative, "Chat" mode, lets the model use its general knowledge too.
4. Click the upload icon and drag your vault's `.md` files into the upload area. Or use the document manager to browse and select files.
5. Click "Move to Workspace" and then "Embed" to vectorize the documents.

Keeping the vault in sync

AnythingLLM's desktop app has a Live Document Sync feature (beta) that watches individual files for changes and re-embeds them every 10 minutes. It's limited: file-level only, no folder watching, requires the desktop app to be open, and only works for files uploaded after version 1.5.9.

For bulk vault sync, use the third-party anythingllm-document-sync Python tool:

```
# Install
git clone https://github.com/dastra/anythingllm-document-sync
cd anythingllm-document-sync
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt

# Configure (~/.anythingllm-sync/config.yaml)
```

```
cat > ~/.anythingllm-sync/config.yaml << 'EOF'
api-key: YOUR_API_KEY
workspace-slug: obsidian-vault
file-paths:
  - /path/to/your/obsidian/vault/
directory-excludes:
  - .obsidian
  - .trash
file-excludes:
  - .DS_Store
EOF

# Run sync
python ingest_anythingllm_docs.py
```

Get your API key from AnythingLLM > Settings > Developer Options > Generate API Key. Get the workspace slug from Workspace Settings > Vector Database tab. Run the sync script via cron for automatic updates.

Obsidian plugin (optional)

There's a community plugin (anything-obsidian) that lets you send individual notes from Obsidian to AnythingLLM workspaces. It's not in the official plugin store — manual install required. It handles one file at a time, so it's useful for pushing specific notes but not for bulk vault sync.

Which models for knowledge work?

The model you pick matters more for note Q&A than for generic chat. Summarizing a 3,000-word research note requires stronger instruction-following than answering "what's the capital of France." Here's what works:

Model	VRAM needed	Speed (approx.)	Best for
Qwen 2.5 7B Q4	~5GB	30-40 tok/s	Quick lookups, tagging, short answers
Qwen 2.5 14B Q4	~10GB	15-25 tok/s	Note summarization, Q&A, synthesis
Qwen 3 30B-A3B Q4	~8GB active	20-35 tok/s	Complex reasoning with MoE efficiency
Command R 35B Q4	~20GB	10-15 tok/s	Long-context RAG, multi-document analysis

Qwen 2.5 14B is the sweet spot for most setups. It handles summarization, Q&A, and multi-note synthesis well, and runs comfortably on 16GB VRAM or a Mac with 24GB unified memory.

Qwen 3 30B-A3B is a mixture-of-experts model — 30B total parameters but only ~3B active per token. It gives you 30B-class reasoning at 7B-class speeds and VRAM usage. If your hardware supports it, it's worth trying for complex synthesis tasks.

Command R 35B was specifically designed by Cohere for RAG workflows. It handles long retrieved contexts well and follows grounding instructions (answering only from provided sources) more reliably than general-purpose models. Needs 24GB VRAM.

For all three paths, the embedding model matters less than the chat model for response quality. `nomic-embed-text` is the standard choice — 8K token context, 274MB, good retrieval accuracy. The built-in models in Smart Connections and AnythingLLM (`multilingual-e5-small` and `all-MiniLM-L6-v2` respectively) work fine for getting started but have 512-token context limits that chop long notes into small pieces.

Practical workflows

Once the infrastructure is running, here's what actually becomes useful day-to-day:

Weekly review. Ask your model "What are the main themes from my notes this week?" It pulls from recent notes and produces a synthesis that's faster than scanning through them manually. Works well in all three setups.

Pre-writing research. Before starting a blog post or report, ask "What have I written about [topic]?" The model retrieves relevant notes across your vault and gives you a starting point. In Smart Connections, use the connections sidebar to discover related notes passively while you write.

Cross-note Q&A. "What did I conclude about model quantization in my GPU benchmarking notes?" The model finds the relevant note, extracts the conclusion, and cites where it came from. This is the core value of RAG over a vault — it replaces 5 minutes of manual searching with a 10-second query.

Contradiction detection. "Find contradictions between my notes on X." Local models handle this reasonably well with 14B+ parameters. Feed it notes that might conflict, and it identifies inconsistencies you'd miss by reading them separately.

Tagging and categorization. Feed new notes through a prompt that suggests tags based on your existing taxonomy. A 7B model handles this at 30+ tok/s, fast enough to tag notes as you create them.

Limitations to know about

Retrieval is not search. RAG retrieves the most semantically similar chunks to your query, not all relevant information. If your vault has 10 notes about a topic, RAG might pull 3-5 of them depending on the Top K setting. It doesn't guarantee comprehensive coverage.

Model size matters for quality. A 7B model will occasionally hallucinate connections between notes that don't exist. 14B is noticeably more reliable. Always check the source citations (Smart Chat and Copilot both show which notes were used).

The sync problem is real. Smart Connections handles this transparently — it re-embeds notes as you edit them. Open WebUI and AnythingLLM require manual or scripted sync, which means your RAG index can be hours behind your actual vault. For reference material this doesn't matter. For active daily notes it's a friction point.

Embedding is a one-time cost, mostly. The initial embedding run is the slow part. After that, only new and modified notes get re-embedded. But if you switch embedding models, you start over from scratch. Pick one and stick with it.

You can't fit your whole vault into context. Even with a 32K context window, you can fit roughly 15-20 notes per query. RAG solves this by selecting only the most relevant chunks, but it's selecting, not searching everything. Quality depends on the embedding model catching the right matches.

Recommendation

Start with Smart Connections. It's inside Obsidian, it indexes automatically, and the connections sidebar alone is worth the install — it surfaces related notes you forgot you wrote. Add Ollama for Smart Chat, and you have a working local AI second brain in 10 minutes.

Move to Open WebUI or AnythingLLM when you need more: multi-model switching, workspace isolation, agent capabilities, or a dedicated chat interface outside Obsidian. Both are solid, both have the vault sync limitation, and both benefit from the same Ollama backend you already set up for Smart Connections.

The point of all three paths is the same: your notes never leave your machine, you pay nothing after the hardware, and the AI works offline. Pick the one that fits your workflow and start using it.

> Check if your GPU can handle your chosen model with the VRAM Calculator. For more on embedding models and RAG quality, see Embedding Models for RAG. And for step-by-step Copilot setup (another excellent Obsidian plugin), see our Obsidian + Copilot guide.

Related guides

- Set Up Local RAG to Search Your Own Documents Privately
- Best Embedding Models for RAG in 2026
- Obsidian + Local LLM via Copilot: Step-by-Step
- Building a Local AI Assistant
- Context Length Explained
- Memory Leak in Long Conversations: Causes and Fixes
