Obsidian + Local LLM: Build a Private AI Second Brain

February 23, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Install Ollama, pull Qwen 2.5 and nomic-embed-text, then add the Copilot plugin in Obsidian. Set Ollama as your provider, point it at localhost:11434, and you have a private AI assistant that can chat with your notes. Your data never leaves your machine.

📚 **More on this topic:** Local RAG Guide · Embedding Models for RAG · Building a Local AI Assistant · VRAM Calculator

Your notes are the most personal data you own. Research, journals, ideas, meeting notes, half-formed theories — all of it sitting in your Obsidian vault. Cloud-based AI tools want you to upload that vault to their servers. That should make you uncomfortable.

The better path: run a local LLM on your own hardware and connect it to Obsidian. Your notes never leave your machine, you pay nothing after the initial hardware, and no API keys leak your data to OpenAI. You get AI-powered search, summarization, and chat across your entire vault, and you keep full control.

This guide walks you through the complete setup, from installing Ollama to chatting with your notes through the Copilot plugin. If you have a GPU with 8GB+ VRAM (or a Mac with 16GB+ unified memory), you can have this running in under 30 minutes.

Why add a local LLM to Obsidian?

Obsidian is already the best local-first knowledge management tool. It stores everything as plain markdown files on your disk. No vendor lock-in, no cloud sync required. Adding a local LLM extends that philosophy to AI. Your intelligence layer runs on the same machine as your notes.

Here is what a local LLM unlocks in Obsidian:

- **Ask questions about your notes.** "What did I write about transformer architectures last month?" You get an actual answer pulled from your vault, not a generic ChatGPT response.
- **Generate summaries.** Select a 3,000-word research note and get a 200-word summary in seconds.

- **Find connections.** Embedding-based semantic search surfaces related notes that keyword search misses entirely.
- **Draft from research.** Highlight five notes on a topic and ask the LLM to draft a blog post outline from them.
- **Auto-tag and categorize.** Run new notes through a prompt that suggests tags based on your existing taxonomy.

The important difference from cloud AI: the model has no memory of your data after you close the session. It does not train on your notes. It does not phone home. This is genuinely private AI.

Plugin comparison: which one should you use?

Three major plugins connect Obsidian to local LLMs. Each has a different strength.

Feature	Copilot	Smart Connections	Text Generator
Primary use	Chat with notes	Semantic search & related notes	Inline text generation
Chat interface	Yes — sidebar panel	Yes — Smart Chat	No (inline only)
Ollama support	Native	Native	Via custom endpoint
Embedding support	Yes (nomic-embed-text)	Yes (built-in + Ollama)	No
RAG over vault	Yes — Relevant Notes feature	Yes — core feature	No
Ease of setup	Easy	Moderate	Moderate
Best for	Conversational Q&A	Discovering note connections	Writing assistance
GitHub stars	7K+	3K+	2K+

Copilot (recommended starting point)

Copilot by Logan Yang is the most polished option for chatting with your notes using a local LLM. It supports Ollama natively, handles CORS issues gracefully (a toggle in settings fixes it), and includes a “Relevant Notes” feature that automatically pulls context from your vault into each conversation. As of 2025, it also supports thinking/reasoning tokens from local models.

If you want one plugin that does 80% of what you need, start here.

Smart Connections

Smart Connections by Brian Petro focuses on embedding-based semantic search. It generates vector embeddings for every note in your vault and uses them to show related notes in a sidebar. It also has a chat feature (Smart Chat) that uses RAG to answer questions from your notes.

The real strength is passive discovery. It constantly shows you notes related to what you are currently working on, even if they share zero keywords. The tradeoff: initial embedding of a large vault takes time (expect 30-60 minutes for 1,000+ notes on a mid-range GPU).

Text Generator

Text Generator by nhaouari is built for inline writing assistance. Select text, run a prompt template, and the output appears right in your note. It connects to Ollama via custom endpoint configuration. Pick this one if you want AI as a writing tool rather than a search or chat tool.

My recommendation: Install Copilot first. Add Smart Connections later if you want semantic search across your vault. Use Text Generator only if inline generation is your primary workflow.

Model recommendations by hardware

The model you pick depends entirely on your GPU. Bigger is not always better. A model that runs at 2 tokens/second is unusable for interactive chat, and you want at least 10 tok/s for a responsive experience.

Your Hardware	Recommended Model	VRAM Usage	Speed (approx.)
8GB VRAM (RTX 3060, 4060)	Qwen 2.5 7B Q4	~5GB	25-35 tok/s
12GB VRAM (RTX 3060 12GB)	Qwen 2.5 7B Q8 or 14B Q4	~8-10GB	15-30 tok/s
16GB VRAM (RTX 4060 Ti 16GB)	Qwen 2.5 14B Q4	~10GB	15-25 tok/s
24GB VRAM (RTX 3090, 4090)	Qwen 2.5 32B Q4	~20GB	10-20 tok/s
Mac M1/M2/M3 16GB	Qwen 2.5 7B Q4	~5GB shared	15-25 tok/s
Mac M4 Pro/Max 36GB+	Qwen 2.5 14B-32B Q4	10-20GB shared	10-30 tok/s

Why Qwen 2.5? It consistently ranks as the best open model family for instruction-following and reasoning at each size tier. For note-assistant tasks like summarization, Q&A, and

categorization, it outperforms Llama 3.1 and Mistral at equivalent sizes. Check our VRAM requirements guide for more detailed numbers.

Embedding model

For semantic search (used by both Copilot and Smart Connections), you need a separate embedding model. The standard choice is **nomic-embed-text**, which is small (~274MB), fast, and produces good embeddings for RAG. Read more about embedding model choices.

Step-by-step setup: Obsidian + Ollama + Copilot

This is the core tutorial. Follow these steps in order and you will have a working local AI assistant inside Obsidian.

Step 1: install Ollama

If you already have Ollama running, skip to Step 2. If not, follow our complete Ollama setup guide or use the quick version below.

Linux:

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

macOS:

Download from ollama.com and drag to Applications.

Windows:

Download the installer from ollama.com and run it.

Verify it is working:

```
ollama --version
```

You should see a version number (0.5.x or higher as of early 2026).

Step 2: pull your chat model

Open a terminal and pull the model that matches your hardware. For this guide, I will use the 14B as the example. Adjust the size based on the hardware table above.

```
ollama pull qwen2.5:14b
```

This downloads roughly 9GB. On a decent connection, expect 5-10 minutes.

If you have only 8GB VRAM:

```
ollama pull qwen2.5:7b
```

If you have 24GB VRAM and want the best quality:

```
ollama pull qwen2.5:32b
```

Verify the model loaded:

```
ollama list
```

You should see your model listed with its size and quantization level.

Step 3: pull the embedding model

This is required for the “Relevant Notes” feature in Copilot and for Smart Connections. It runs alongside your chat model.

```
ollama pull nomic-embed-text
```

This is a small download (~274MB). It takes seconds.

Step 4: start the Ollama server

On macOS, the Ollama app starts the server automatically when you launch it. On Linux and Windows, the installer typically sets up a background service.

Check that the server is running:

```
curl http://localhost:11434
```

You should see: `Ollama is running`

If the server is not running, start it manually:

```
ollama serve
```

Important for Linux/Windows users: If you hit CORS errors later when Obsidian tries to connect, start the server with the Obsidian origin allowed:

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

On macOS, you can set this as an environment variable:

```
launchctl setenv OLLAMA_ORIGINS "app://obsidian.md*"
```

Then restart the Ollama app.

Step 5: install Copilot in Obsidian

1. Open Obsidian and go to **Settings** (gear icon, bottom-left).
2. Click **Community Plugins** in the left sidebar.
3. If community plugins are disabled, click **Turn on community plugins** and confirm.
4. Click **Browse** to open the community plugin browser.
5. Search for **“Copilot”** and look for “Copilot” by Logan Yang (it will have the most downloads).

6. Click **Install**, then **Enable**.

You will now see a Copilot chat icon in your left sidebar.

Step 6: configure Copilot for Ollama

1. Go to **Settings > Copilot** (under Community Plugins in the left sidebar).
2. Under the **Model** section, click **Add Custom Model**.
3. Fill in:
   - **Model Name:** `qwen2.5:14b` (must match exactly what `ollama list` shows)
   - **Provider:** Select **ollama**
   - **Base URL:** Leave as default (`http://localhost:11434`) or enter it if blank
4. Click **Add** to save the model.
5. Back in the main settings, select your new model from the model picker dropdown.
6. If you see CORS errors in the Copilot chat, go to the model table in Copilot settings and toggle the **CORS** switch on for your Ollama model. This routes requests through a proxy that avoids browser CORS restrictions.

Step 7: add the embedding model

Still in Copilot settings:

1. Scroll down to **QA Settings** (or Embedding Settings, depending on version).
2. Set the **Embedding Model** to `nomic-embed-text`.
3. Set the **Embedding Provider** to **ollama**.
4. Click **Build Index** (or the refresh icon) to start indexing your vault.

Indexing time depends on vault size:

- 100 notes: under 1 minute
- 500 notes: 2-5 minutes
- 2,000 notes: 10-20 minutes
- 5,000+ notes: 30-60 minutes (first run only, incremental after)

Step 8: chat with your notes

1. Click the Copilot icon in the left sidebar to open the chat panel.
2. Type a question like: “Summarize what I know about local LLMs”

3. Copilot will search your vault for relevant notes, inject them as context, and generate a response grounded in your actual notes.

You will see a **Relevant Notes** section at the top of each response showing which notes were used as context. Click any to jump directly to that note.

Pro tip: Use the `@` symbol to reference specific notes in your prompt. Type `@` followed by a note name to explicitly include it as context. This is useful when you want the model to focus on specific source material.

Advanced: full vault RAG with Smart Connections

Copilot's Relevant Notes feature is solid for chat-based Q&A. But if you want always-on semantic search, a sidebar that constantly shows notes related to what you are looking at, add Smart Connections.

Install and configure Smart Connections

1. Go to **Settings > Community Plugins > Browse**.
2. Search for **"Smart Connections"** and install it.
3. In **Settings > Smart Connections**:
   - Set **Embedding Model Platform** to **Ollama (Local)**.
   - Set **Embedding Model** to `nomic-embed-text:latest`.
   - Set the connection details: protocol `http`, hostname `localhost`, port `11434`.
4. For Smart Chat, set the **Chat Model Platform** to **Ollama** and select your chat model (e.g., `qwen2.5:14b`). The path should be `/api/chat`.
5. Click **Build Embeddings** or let it run automatically.

How Smart Connections works

Smart Connections splits every note in your vault into chunks, generates a vector embedding for each chunk using nomic-embed-text, and stores the embeddings locally. When you open a note, it calculates similarity scores against every other chunk in your vault and displays the most semantically related notes in the **Smart View** sidebar.

This catches connections that Obsidian's built-in backlinks and search miss entirely. Two notes about "model quantization" and "reducing inference memory" will surface as related even if they share no common keywords.

Smart Chat

Open Smart Chat from the command palette or sidebar. Ask questions like:

- "What are the main arguments in my notes about privacy?"
- "Find contradictions between my notes on model benchmarks"
- "What research have I collected about RAG chunking strategies?"

Smart Chat performs RAG (retrieval-augmented generation). It retrieves the most relevant note chunks via embedding similarity, feeds them to the local LLM as context, and generates a grounded response.

Performance tips

Running a local LLM alongside Obsidian uses significant system resources. Here is how to keep things snappy.

Use the right model size for the task

You do not need a 32B model for everything. A practical approach:

- **Quick categorization, tagging, short answers:** Qwen 2.5 7B. Fast responses, low VRAM.
- **Summarization, Q&A over notes:** Qwen 2.5 14B. Good balance of quality and speed.
- **Complex analysis, multi-note synthesis:** Qwen 2.5 32B. Better reasoning, slower.

Copilot lets you add multiple Ollama models and switch between them in the model picker. Pull two sizes and swap depending on the task.

Context window settings

Ollama defaults to a 2048-token context window for most models. For note Q&A, you want more context to fit relevant note excerpts. Increase it:

```
ollama run qwen2.5:14b
/set parameter num_ctx 8192
/bye
```

This sets the context window to 8,192 tokens (~6,000 words of context). For summarization tasks, you may want 16384 or even 32768 if your VRAM allows it. Each doubling of context roughly doubles the memory needed for the KV cache.

Keep embedding and chat models loaded

Ollama unloads models after 5 minutes of inactivity by default. If you are switching between Obsidian and other work, the first query after a pause will be slow while the model reloads. Increase the keep-alive time:

```
OLLAMA_KEEP_ALIVE=30m ollama serve
```

This keeps models in memory for 30 minutes after the last request.

Hardware-specific notes

NVIDIA GPU users: Make sure your Ollama installation is using your GPU. Run `nvidia-smi` while a model is loaded and you should see `ollama_llama_server` in the process list using VRAM.

Mac users: Ollama automatically uses Metal acceleration on Apple Silicon. Unified memory means the model and Obsidian share the same memory pool. With 16GB total, a 7B model runs comfortably. With 36GB+, 14B-32B models work well.

CPU-only users: You can run 7B models on CPU, but expect 2-5 tok/s. Usable for short queries, painful for anything longer. A used GPU with 8GB+ VRAM is a worthwhile $100-150 investment. See our guide on used GPUs for local AI.

Limitations and honest expectations

Local LLMs inside Obsidian are genuinely useful. They are not magic. Here is what to expect.

Accuracy is model-dependent. A 7B model will occasionally hallucinate connections between notes that do not exist. Larger models are more reliable but still imperfect. Always verify claims against the source notes Copilot references.

Large vaults need chunking strategy. If you have 10,000+ notes, embedding everything takes time and the retrieval quality depends heavily on chunk size. Smart Connections defaults to

splitting by headings, which works well for structured notes. For long-form notes without headers, consider adding headers to improve chunking.

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

You cannot fit your whole vault into context. Even with a 32K context window, you can only fit roughly 15-20 notes worth of content per query. RAG solves this by selecting only the most relevant chunks, but it is not searching your entire vault on every query. It searches the embedding index and pulls top matches.

No internet, no problem. Once Ollama and your models are downloaded, everything works offline. Airplane mode, no Wi-Fi, deep in the mountains, your AI second brain still works. That is the entire point.

Recommended workflow

After a week of using this setup, here is the workflow that stuck:

1. **Daily notes and research** go into Obsidian as always. Plain markdown, light tags.
2. **Weekly review:** Open Copilot and ask "What are the main themes from my notes this week?" It pulls from recent notes and gives a surprisingly good synthesis.
3. **Before writing:** Select 5-10 source notes with `@` mentions and ask the model to draft an outline. Faster than staring at a blank page.
4. **Passive discovery:** Keep the Smart Connections sidebar open. While working on a note about one topic, you will see related notes from months ago that you forgot existed.
5. **Quick lookups:** Instead of searching with keywords (which misses synonyms and related concepts), ask Copilot in natural language.

This does not replace your own thinking. But if you spend 20 minutes hunting through old notes before you start writing, that drops to about 30 seconds with Copilot. The time savings compound fast once you have a few hundred notes indexed.

Related guides

- Set Up Local RAG to Search Your Own Documents Privately

- Best Embedding Models for RAG in 2026
- Building a Local AI Assistant That Runs Entirely on Your Hardware
- Ollama vs LM Studio: Which Local LLM Runner Should You Use?
