**5 self-hosted LLMs I use for specific tasks**

Story by Yash Patel

For the past few years, my world of AI was limited to the big names like ChatGPT and Gemini. But when my colleagues at XDA started sharing their [experiences with self-hosted LLMs](https://www.xda-developers.com/i-started-self-hosting-llms-absolutely-loved-it/?utm_source=syndication&pubDate=20250810), my curiosity was piqued. The idea of running a powerful AI on my own machine, completely under my control, was incredibly appealing to me. So, I decided to give it a try.

I set up Ollama and the Open WebUI in Docker on my laptop, which is equipped with an Intel Core Ultra 9 processor, 32GB of RAM, and an NVIDIA GeForce RTX 4050 GPU. This powerful local setup became my playground, allowing me to explore and experiment with various LLMs. Instead of a one-size-fits-all approach, I discovered that different models excel at different tasks. After trying dozens of models, I incorporated these five self-hosted LLMs into my daily workflow.

**qwen2.5-coder**

**Your go-to coding companion**



Install qwen coder llm on local machine using Ollama

Once a coder, always a coder! Even though I don't code full-time now, I still enjoy it. I love jumping into competitive programming or writing scripts to make my life easier. When I decided to run an AI model on my own computer, I wanted a dedicated LLM for coding. That's why I chose qwen2.5-coder. It's a specialized LLM from the same family as qwen 2.5, but it's fine-tuned specifically for coding tasks. It is trained on an enormous dataset of code, providing it with a deep understanding of over 40 programming languages.

I got it running on my Windows 11 laptop using Ollama and a WebUI inside Docker. This model is specially trained on a huge amount of code. The qwen-coder2.5 comes in different sizes, from a small 0.5B model to a powerful 32B model. Due to hardware limitations, I picked the 7B model. It is perfect for writing new code, fixing bugs, and repairing broken code. It's a huge help for troubleshooting. Whether I'm facing a simple syntax error or a more complex logical bug, it’s my go-to coding companion. I use it to create scripts that [automate various tools I use daily](https://www.xda-developers.com/i-use-automation-to-supercharge-logseq-journaling/?utm_source=syndication&pubDate=20250810).

**wizard-math**

**Your logic partner**



wizard-math home page in WebUI

I was exploring different LLMs on Ollama and found wizard-math. As someone who loves a good challenge, especially mathematical puzzles and logical reasoning, I thought of giving a dedicated LLM built for math and reasoning a try. I quickly self-hosted wizard-math on my Ollama + WebUI setup.

I set it up for fun and to expand my own knowledge. This model is a specialized version of the WizardLM family. It is trained to excel in complex mathematical problems, logical reasoning, and solving puzzles. There are three different sizes available for wizard-math – 7B, 13B, and 70B. With my little experience exploring wizard-math with 7B, I found it fantastic. It helps me test my solutions and explore new ways to approach difficult problems. The model's ability to handle these subjects with precision and clarity makes it my go-to partner for all things logic and numbers.

**reader-lm**

**Web to markdown, instantly**



reader-lm displayed in cmd and WebUI

I have [connected Obsidian and Logseq to improve my daily workflow](https://www.xda-developers.com/connect-obsidian-and-logseq-best-of-both/?utm_source=syndication&pubDate=20250810), and markdown files are central to my entire setup. All my notes and research are in this format. The reader-lm is a special LLM built for this specific task to convert HTML content into a clean markdown file. To easily handle content from the web, I self-hosted reader-lm.

The model is super practical for my needs. Instead of manually creating *.md* files from the web content, I can feed them to reader-lm to get a perfectly structured markdown file. In my experience, while reader-lm does an amazing job for most of my needs, it sometimes struggles with really large or messy HTML code. But, it works well enough most of the time and makes a huge difference in my productivity. It ensures my [notes are consistent](https://www.xda-developers.com/i-prefer-logseq-over-obsidian-for-daily-journaling/?utm_source=syndication&pubDate=20250810) and easy to read. It's a great example of using a specialized LLM to [automate a simple but time-consuming task](https://www.xda-developers.com/combined-notion-with-n8n-automate-bookmarking/?utm_source=syndication&pubDate=20250810).

**llma-guard3**

**An LLM for safe prompts**



llma-guard3 self-hosted on laptop example

When working with LLMs, it’s crucial that our interactions are safe and responsible. While we can’t control an LLM’s response, we can ensure our prompts are appropriate. That’s exactly why I self-hosted llama-guard 3. This powerful model acts as a dedicated content moderation tool for all my other local LLMs.

The llama-guard 3's job is to classify every interaction against a set of safety categories. It checks our prompts for 13 different categories. When we give a prompt to this LLM, it will respond with a message stating whether the prompt message was safe or unsafe. If it is unsafe, it flags it with a specific reason, such as S1 (Hate Speech) or S2 (Sexual Content), etc.

**Gemma 3**

**My local Gemini experience**



gemma3 sample answer in cmd

ChatGPT and Gemini are two key benchmarks that made everyone accustomed to AI and LLMs. While self-hosting LLMs, I also did not want to compromise my experience with those platforms. That’s why I self-hosted Gemma 3. This model is built on the same research as Gemini. It provides a premium experience with the flexibility of running locally. It is basically my local ChatGPT / Gemini.

Gemma 3 is available in various sizes. It can handle a massive 128k context window, processes both text and images, and understands over 140 languages. This makes it my personal go-to AI for creative tasks. I use it to generate ideas for social media content, draft captions, and research topics for my blog.

**Hybrid approach is the key**

I now have dedicated, powerful AI assistants running on my laptop, which give me full control over my data and privacy. This [setup is definitely productive](https://www.xda-developers.com/productivity-workloads-made-easier-with-local-llm/?utm_source=syndication&pubDate=20250810) for specific tasks, but based on my experience, I've also learned that I can't completely rely on it. There are occasional speed and reliability issues. For more complex or important tasks, I still use cloud-based services like ChatGPT or Gemini. The real power is in using a hybrid approach, leveraging the privacy of local models while still having access to the cutting-edge capabilities of commercial services.