Local AI for Privacy: What's Actually Private

February 10, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Running AI locally with Ollama, LM Studio, or llama.cpp keeps your prompts, documents, and usage patterns completely off corporate servers. No one trains on your data, no one reads your conversations, and you don't even need an account. But 'local' doesn't mean 'automatically private.' Model downloads expose your IP to Hugging Face. Some tools check for updates on launch. VS Code AI extensions have been caught exfiltrating code to China. And ComfyUI's desktop app shipped rogue telemetry that ignored the off switch. True privacy requires knowing what your tools actually do on the network — then shutting off what you don't need.

📚 **Related:** Running AI Offline · Ollama vs LM Studio · LM Studio Tips · OpenClaw Security Guide

“Run it locally and your data stays private.”

You’ve seen this on every Reddit thread, every Hacker News comment, every local AI tutorial. And it’s mostly true. But “mostly” is doing a lot of work when the reason you went local was to keep sensitive documents away from corporate servers.

Here’s the actual picture: what’s genuinely private, what still leaks, and how to lock it all down.

What’s Genuinely Private

When you run a model locally with Ollama, LM Studio, or llama.cpp, these things stay on your machine with zero exceptions:

Your prompts. Every question, every document you paste, every conversation — none of it leaves your hardware. No API call, no round trip to a datacenter, no server-side log.

Your responses. The model generates output using your CPU or GPU. No copy exists on anyone else’s infrastructure.

Your documents. When you use local RAG to search your files, those files stay on your disk. Embeddings are generated locally. The vector database lives on your machine.

Your usage patterns. No one knows what models you run, when you run them, how often, or what topics you explore. There's no analytics dashboard on someone else's server tracking your behavior.

No account required. Ollama doesn't need a login. LM Studio doesn't need a login. llama.cpp definitely doesn't need a login. You can run inference without ever giving anyone your email address.

Works offline. Once you've downloaded a model, disconnect from the internet entirely and everything still works. That's air-gapped privacy — the kind you can't get from any cloud service.

Compare that to what happens when you use cloud AI:

	Local AI	ChatGPT (Free)	Claude (Free/Pro)	Gemini (Free)
Prompts sent to servers	No	Yes	Yes	Yes
Trains on your data (default)	No	Yes	Yes (since Sep 2025)	Yes
Can opt out of training	N/A	Yes	Yes	Yes
Human reviewers can see chats	No	Yes	Possible	Yes
Data retention	You control	30 days	Up to 5 years	Up to 3 years
Account required	No	Yes	Yes	Yes
Works offline	Yes	No	No	No

The bottom line: local AI eliminates the single biggest privacy risk — sending your data to a third party who stores it, trains on it, or lets employees read it.

What's NOT Automatically Private

Here's where the "just run it locally" advice falls short. Running locally solves the biggest problem, but these gaps still exist.

Model Downloads Expose Your IP

Every model you pull contacts a server. `ollama pull llama3.2` hits registry.ollama.ai. Downloading from Hugging Face hits huggingface.co. The hosting provider sees your IP address and which model you're downloading.

For most people this doesn't matter. But if you're trying to keep your AI usage invisible, plan your downloads separately from your inference work — use a VPN, download on a different network, or transfer models via USB from another machine.

Telemetry and Update Checks

Ollama contacts registry.ollama.ai on startup for update checks. The core runtime has no telemetry — your prompts never leave your machine. But the update check reveals your IP and that you're running Ollama. There's no built-in flag to disable this yet (it's an open feature request). The workaround is to block registry.ollama.ai in your firewall or `/etc/hosts`.

LM Studio contacts its servers when you search for models, download models, or check for software updates. Outside of those three actions, it makes no outbound connections. No analytics, no tracking, no login required.

ComfyUI's desktop app had a telemetry problem. Version 0.4.41 fixed "rogue remote telemetry" that was sending activity data even when users had toggled the setting off. The core open-source ComfyUI code has no telemetry. If you use the desktop app, update to the latest version and verify the toggle is off.

llama.cpp makes zero outbound connections. Period. No update checks, no telemetry, no network code in the inference engine. If you want the gold standard for verified-private inference, this is it.

Web Search Features Phone Home

Some local AI front-ends include web search — Open WebUI can search the web to augment responses, and various RAG tools include web retrieval. The moment web search activates, your queries leave your machine and hit search APIs. This is by design, but it breaks the air-gap.

If privacy is the goal, disable web search features or use them only for non-sensitive queries.

VS Code AI Extensions: The Real Threat

This one is worse than telemetry. In January 2026, researchers discovered two VS Code extensions masquerading as AI coding assistants — "ChatGPT – 中文版" and "ChatMoss (CodeMoss)" — that had been installed by **1.5 million developers**. These extensions:

- Captured every file you opened in VS Code, not just files you interacted with
- Base64-encoded the contents and sent them to servers in China
- Could remotely trigger harvesting of up to 50 files from your workspace
- Embedded hidden tracking iframes with four Chinese analytics SDKs

- Exfiltrated `.env` files, API keys, SSH keys, credentials, and source code

The extensions actually worked as AI assistants, which is why the malicious behavior went undetected for months.

This wasn't a one-off. Malicious VS Code extension detections grew from 27 in all of 2024 to 105 in the first ten months of 2025. The VS Code marketplace has a real supply chain security problem.

The lesson: Your local LLM can be perfectly private, but if you've installed a sketchy VS Code extension alongside it, your code is leaking anyway. Audit your extensions. Check publishers. Remove anything you don't actively use.

"Local" Tools with Cloud Fallbacks

Some tools that market themselves as "local" include cloud API fallbacks. If the local model fails or takes too long, they silently route your prompt to a cloud endpoint. Read the docs. Check the settings. If a tool supports multiple "providers" including cloud APIs, make sure the cloud option is disabled — not just not-selected, but actually disabled.

Tools That Respect Privacy

Here's what you can trust and what to watch for:

Tool	Telemetry	Update Checks	Network During Inference	Privacy Verdict
llama.cpp	None	None	None	Best possible
Ollama	None	Yes (startup)	None	Great, block update checks for max privacy
LM Studio	None	Yes (launch)	None	Great, download models manually for max privacy
ComfyUI (core)	None	None	None	Great for local image gen
ComfyUI (desktop)	Opt-in (had a bug)	Yes	None	Good, verify telemetry toggle
Open WebUI	None	None	Only if web search enabled	Good, disable web search for privacy

For maximum privacy: llama.cpp with a pre-downloaded GGUF model is unbeatable. Zero network code means zero possible data leakage during inference. The tradeoff is a command-line interface and manual setup.

For practical privacy: Ollama with update checks blocked gives you an easy-to-use tool with no data leakage. Add a front-end like Open WebUI bound to localhost and you get a ChatGPT-like interface that's genuinely private.

Tools to Audit

Before trusting any tool with sensitive work, check for these:

Cloud sync features. If a note-taking app or RAG tool offers "sync across devices," your data is hitting a server. Turn it off or use a tool that doesn't have it.

Update checks on launch. Most GUI tools ping a server on startup. This reveals your IP and that you're using the tool. Usually harmless, but block it if you need to.

RAG tools with built-in web search. AnythingLLM, Open WebUI, and others can search the web during retrieval. That's useful, but it means your questions leave your machine. Disable it for sensitive queries.

VS Code extensions. After the MaliciousCorgi incident, treat every AI extension as suspect until proven otherwise. Check:

- Is the publisher verified?
- Is the source code available?
- Does it request network permissions?
- When was it last updated?
- What permissions does it request?

GitHub Copilot. On the free tier, your code may be used for model improvement. Business and Enterprise plans don't train on your data, but your code still goes to GitHub's servers for inference. If the code can't leave your machine, Copilot isn't an option — use a local coding model instead.

Your Threat Model Matters

Not everyone needs the same level of privacy. What are you actually protecting against?

Hiding from Big Tech Data Collection

Threat: You don't want OpenAI, Google, or Anthropic training on your prompts, storing your conversations, or letting employees read them.

Solution: Any local setup works. Even Ollama with default settings keeps your prompts completely off corporate servers. This is the easiest threat to address and the most common reason people go local.

Hiding from Network Snoopers

Threat: Someone on your local network (IT department, shared WiFi, ISP) can see your traffic patterns.

Solution: They can see that you downloaded models from Hugging Face, but they can't see your prompts or responses (those never leave your machine). For model downloads, use a VPN. For inference, there's nothing to intercept — it's all local computation. Enable full disk encryption so your data is protected if the machine is physically accessed.

Hiding from Nation-State Adversaries

Threat: Advanced persistent threats, hardware implants, supply chain compromises.

Solution: Different conversation entirely. Air-gapped hardware, verified firmware, hardware security modules, TEMPEST shielding. If this is your threat model, you're not reading a blog post for advice — you have a security team. But the fundamentals still apply: local inference on air-gapped hardware with pre-loaded models is as good as it gets for the AI layer.

Practical Privacy Setup

Five steps to lock down a local AI setup. Takes about 15 minutes.

1. Use Ollama + Terminal (Minimal Attack Surface)

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

# Pull a model while online
ollama pull qwen2.5:14b

# Bind to localhost only
export OLLAMA_HOST=127.0.0.1:11434
```

The terminal has the smallest attack surface of any interface. No browser extensions, no Electron app, no hidden iframes.

2. Pre-Download Models, Then Work Offline

```
# Download everything you need while connected
ollama pull qwen2.5:14b
ollama pull llama3.2:3b
ollama pull nomic-embed-text

# Disconnect from the internet
# On Linux:
nmcli networking off

# Now run inference — everything works
ollama run qwen2.5:14b
```

See our complete offline guide for portable setups and model transfer via USB.

3. Disable Telemetry in GUI Tools

LM Studio: Download models manually from Hugging Face and load from disk. This avoids the model search network call entirely.

ComfyUI Desktop: Settings → verify "Send anonymous usage metrics" is toggled off. Update to 0.4.41+ to fix the rogue telemetry bug.

Open WebUI: Disable web search in settings if you don't need it.

4. Firewall Outbound Connections During Use

```
# Block Ollama's update checks
echo "127.0.0.1 registry.ollama.ai" | sudo tee -a /etc/hosts

# Or use ufw to block all outbound during sensitive work
sudo ufw default deny outgoing
sudo ufw enable

# When you need internet again
sudo ufw default allow outgoing
```

For a more surgical approach, block outbound per-application using iptables:

```
# Block outbound for a specific user running Ollama
sudo iptables -A OUTPUT -m owner --uid-owner ollama -d 127.0.0.1 -j ACCEPT
sudo iptables -A OUTPUT -m owner --uid-owner ollama -j DROP
```

5. Full Disk Encryption

If someone gets physical access to your machine, all the local inference in the world doesn't help if your disk is unencrypted.

- **Linux:** LUKS (set up during OS install, or use `cryptsetup`)
- **macOS:** FileVault (System Preferences → Security & Privacy)
- **Windows:** BitLocker (Pro/Enterprise) or VeraCrypt (Home)

This protects your model files, conversation logs, RAG databases, and any documents you've been working with.

When Local Is Non-Negotiable

Some data should never touch a cloud AI service, regardless of their privacy policy:

Legal documents. Client communications, contracts, case files. Attorney-client privilege doesn't survive sending documents to OpenAI's servers, even with training opt-out.

Medical notes. Patient records, clinical notes, diagnostic discussions. HIPAA doesn't care that you opted out of training — the data still left your control.

Proprietary code. Trade secrets, unreleased features, security-sensitive implementations. Use a local coding model or self-host behind your firewall.

Personal journals and private writing. Anything you'd be uncomfortable seeing in a data breach notification. Cloud providers get breached. Local machines only get breached if someone targets you specifically.

Financial data. Tax documents, bank statements, investment strategies. Cloud AI terms of service are not a substitute for financial data security.

When Cloud Is Fine

Not everything needs a privacy fortress. Use cloud AI freely for:

- **General knowledge questions.** "How does TCP handshaking work?" isn't sensitive.
- **Public code help.** Debugging open-source code that's already on GitHub.
- **Non-sensitive chat.** Brainstorming blog post ideas, getting recipe suggestions.
- **Anything you'd post publicly.** If you'd put it on Stack Overflow or Reddit, it's fine to put in ChatGPT.

The right approach is to match the tool to the sensitivity. Use ChatGPT or Claude for casual questions where convenience matters. Switch to local for anything sensitive. The tiered AI model strategy covers this in detail.

The Bottom Line

Local AI gives you genuine privacy that no cloud service can match, no matter what their terms of service say. Your prompts stay local. Your documents stay local. Nobody trains on your data.

But "local" isn't a magic word. Update checks phone home. Model downloads expose your IP. VS Code extensions have exfiltrated code from 1.5 million developers. ComfyUI shipped telemetry that ignored its own off switch.

The fix is straightforward: know what your tools do on the network, block what you don't need, and match your setup to your actual threat model. For most people, Ollama with update checks blocked is plenty. For sensitive professional work, go offline during inference. For regulated data, air-gap the machine.

Start with Ollama, pick a model that fits your hardware, and you're already in a fundamentally better privacy position than 99% of AI users.

Related Guides

- Running AI Offline: Complete Guide
- Ollama vs LM Studio
- LM Studio Tips & Tricks
- llama.cpp vs Ollama vs vLLM
- Local RAG Guide
- Best Local LLMs for Coding
- Local LLMs vs ChatGPT
- OpenClaw Security Guide
- Tiered AI Model Strategy
- Local AI Planning Tool — VRAM Calculator

Source: https://insiderllm.com/guides/local-ai-privacy-guide/