Running AI Offline: Complete Guide to Air-Gapped Local LLMs

February 8, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Ollama, LM Studio, and llama.cpp all work completely offline once you've downloaded a model. Pull your models while connected (`ollama pull qwen2.5:14b`), disconnect from the internet, and everything keeps running — no API calls, no authentication checks, no telemetry required. The minimal offline setup is Ollama + one 7-14B model + 8GB RAM. For a portable kit, load an external SSD with your models and software. The only things that break offline are web search tools, cloud API calls, and model downloads.

📚 **Related:** Run Your First Local LLM · LM Studio Tips · Laptop vs Desktop for AI · Mac M-Series Guide · Planning Tool

The entire point of local AI is running on your hardware. But most guides assume you're online — downloading models mid-tutorial, pulling Docker images, fetching Python packages. What happens when you unplug?

Everything still works. Ollama doesn't phone home. llama.cpp doesn't need a license server. Your models are files on your disk, and inference is pure math on your CPU or GPU. No network required.

But getting to that point — having everything you need pre-downloaded and verified — takes preparation. This guide covers how to set up a fully offline AI stack, what works without internet, what breaks, and how to build a portable kit you can take anywhere.

What Works Offline (And What Doesn't)

Fully Offline After Setup

Tool	Offline?	Notes
Ollama	Yes	No network calls during inference
LM Studio	Yes	Fully local, no account required
llama.cpp	Yes	Bare metal, zero dependencies at runtime

Tool	Offline?	Notes
Text Generation WebUI	Yes	Web UI runs locally on localhost
ComfyUI / Automatic1111	Yes	Image generation, fully local
AnythingLLM	Yes	RAG works offline once documents are embedded
Open WebUI	Yes	Chat interface, runs in Docker locally
ChromaDB / LanceDB	Yes	Vector databases store locally

Breaks Without Internet

Feature	Why It Breaks
Model downloads	Obviously — files need to transfer
Web search tools	Search requires internet connectivity
Cloud API fallbacks	Any tool with cloud routing (check your configs)
Fetching web pages for RAG	Can't scrape URLs offline
Docker image pulls	Need pre-pulled images
Python package installs	`pip install` needs PyPI access
Update checks	Some tools ping servers on launch

The pattern: **inference is local, acquisition is not.** Everything that runs models works offline. Everything that fetches data doesn't. Your job is to do all the fetching before you disconnect.

Preparing for Offline Use

Step 1: Download Your Models

This is the big one. Models are multi-gigabyte files that need to be on disk before you go offline.

Via Ollama:

```
# Pull your daily driver
ollama pull qwen2.5:14b
```

```
# Pull a small model for low-resource situations
ollama pull qwen2.5:7b

# Pull an embedding model for RAG
ollama pull nomic-embed-text

# Verify everything downloaded
ollama list
```

Via LM Studio: Open LM Studio → Search → Download models while connected. They're stored in `~/.cache/lm-studio/models/` (Linux/Mac) or `C:\Users\<you>\.cache\lm-studio\models\` (Windows).

Via llama.cpp: Download GGUF files directly from HuggingFace. Save them to a known directory.

How much disk space?

Model	Approximate Size
7B at Q4	~4-5 GB
14B at Q4	~9 GB
32B at Q4	~20 GB
70B at Q4	~40 GB
Embedding model	~275 MB

A solid offline library: one 14B + one 7B + embedding model = ~14GB. Add a 32B model and you're at ~34GB. Even a modest SSD handles this easily.

Step 2: Test Offline Before You Need It

Don't assume it works — verify it while you still have internet to fix problems.

```
# Linux: Disable networking
nmcli networking off

# Mac: Turn off Wi-Fi from menu bar or
networksetup -setairportpower en0 off
```

```
# Windows: Airplane mode, or
# Settings → Network & Internet → turn off Wi-Fi and Ethernet
```

Now test:

```
# Start Ollama (should work without network)
ollama serve

# Run a model
ollama run qwen2.5:14b
# Type a prompt. If you get a response, you're offline-ready.

# Test embedding model too
ollama run nomic-embed-text
```

What to check for:

- Model loads without errors
- Responses generate at normal speed
- No “connection refused” or “network unreachable” errors in logs
- UI tools (Open WebUI, AnythingLLM) launch and function

```
# Re-enable when done testing
nmcli networking on        # Linux
networksetup -setairportpower en0 on   # Mac
```

Step 3: Pre-Download Dependencies

If you use Python-based tools, download packages in advance:

```
# Download packages without installing (for later offline install)
pip download -d ./offline-packages torch transformers sentence-transformers

# Install later from local cache
pip install --no-index --find-links=./offline-packages torch transformers sentence-transformers
```

Docker images:

```
# Pull while online
docker pull ghcr.io/open-webui/open-webui:main
docker pull mintplexlabs/anythingllm

# These images are cached locally — Docker runs them offline
```

Ollama itself: If you're setting up a new machine offline, download the Ollama installer while connected:

```
# Save the install script
curl -fsSL https://ollama.com/install.sh -o ollama-install.sh

# Or download the binary directly from GitHub releases
# https://github.com/ollama/ollama/releases
```

Offline RAG: Chat With Documents Without Internet

RAG (Retrieval Augmented Generation) works fully offline as long as both your embedding model and chat model are local.

The Stack

1. **Embedding model:** nomic-embed-text via Ollama (~275MB)
2. **Vector database:** ChromaDB or LanceDB (both store locally)
3. **Chat model:** Any Ollama model
4. **Interface:** AnythingLLM (easiest) or custom Python

Setup While Online

```
# Pull models
ollama pull qwen2.5:14b
ollama pull nomic-embed-text

# Install AnythingLLM (desktop app — no Docker needed)
# Download from anythingllm.com
```

```
# Configure AnythingLLM:
# LLM Provider: Ollama (localhost:11434)
# Embedding Provider: Ollama → nomic-embed-text
# Vector DB: LanceDB (built-in, default)

# Upload and embed your documents while online
# (embedding is CPU/GPU intensive but doesn't need internet)
```

Going Offline

Once documents are embedded and models are downloaded, disconnect. Everything works:

- Upload new local documents (file system access doesn't need internet)
- Generate embeddings for new documents (local embedding model)
- Chat with existing document collections
- Create new workspaces

The only thing you can't do is scrape web URLs for RAG input. Download any web content as HTML or PDF before going offline.

For a deeper dive on RAG, see the local RAG guide.

Offline Image Generation

Stable Diffusion and Flux run entirely locally. The preparation is the same: download everything first.

What to Download

Component	Size	Where
SDXL checkpoint	~6.5 GB	HuggingFace or CivitAI
Flux Dev	~12 GB	HuggingFace
VAE (if separate)	~300 MB	Bundled with most checkpoints
LoRAs (optional)	50-200 MB each	CivitAI or HuggingFace
ControlNet models	~1.5 GB each	HuggingFace

ComfyUI Offline Setup

```
# While online: install ComfyUI and download models
git clone https://github.com/comfyanonymous/ComfyUI
cd ComfyUI
pip install -r requirements.txt

# Place models in the correct directories:
# models/checkpoints/    — main model files
# models/loras/          — LoRA files
# models/controlnet/     — ControlNet models
# models/vae/            — VAE files

# Test offline: disable network, launch ComfyUI
python main.py
# Open http://localhost:8188
```

ComfyUI's custom node manager tries to check for updates on launch. This produces warnings offline but doesn't prevent operation. The core generation pipeline is fully local.

Portable Offline Kit

The External SSD Setup

A 500GB-1TB external SSD can hold your entire AI stack — models, software, and documents. Plug it into any machine and run.

What to put on it:

Component	Size	Purpose
Ollama binary	~100 MB	Inference engine
2-3 GGUF models	15-30 GB	Chat, coding, embedding
AnythingLLM AppImage	~200 MB	RAG interface
Document collection	Variable	Your files for RAG
Python + packages	2-3 GB	For custom scripts
SD checkpoint + LoRAs	7-15 GB	Image generation

Component	Size	Purpose
Total	**25-50 GB**	Fits on any SSD

Set the model path to the external drive:

```
# Point Ollama at the external drive
export OLLAMA_MODELS=/mnt/external-ssd/ollama-models
ollama serve
```

Laptop Recommendations

Platform	Best For	Battery Life	Notes
MacBook M1/M2/M3/M4	Battery + performance	8-15 hours	Unified memory, no GPU needed
Gaming laptop (3060+)	Raw speed	2-4 hours	Fast GPU, short battery
ThinkPad + eGPU	Flexibility	6-10 hours (without eGPU)	Portable + powerful when docked

For travel: A MacBook with M-series silicon is the best offline AI device. Unified memory means 16-24GB is directly available for models without a discrete GPU. Battery life stays reasonable even under inference load. A 7B model runs at 20-30 tok/s on an M2 Pro.

Recommended offline models for laptop use:

Model	RAM Needed	tok/s (M2 Pro)	Use Case
Qwen 2.5 7B Q4	~5 GB	~25-30	General chat, writing
Phi-4-mini 3.8B Q4	~3 GB	~40-50	Math, light tasks, battery-saving
Qwen 2.5 14B Q4	~9 GB	~15-18	Best quality if RAM allows

Stick to 7B-14B for battery life. Larger models drain power faster because they move more data through memory per token.

Gotchas and Edge Cases

Telemetry and Phone-Home Behavior

Most local AI tools don't send telemetry, but check:

- **Ollama:** No telemetry. Fully offline-safe.
- **LM Studio:** No telemetry. Offline-safe.
- **Open WebUI:** Checks for updates on launch. Produces a warning offline, doesn't affect function.
- **AnythingLLM:** Checks for updates. Works fine offline with a brief delay on launch.
- **ComfyUI Manager:** Checks for node updates. Creates warnings but doesn't block generation.

If you're in a strict air-gapped environment, monitor outbound network connections during initial setup to verify nothing phones home. Use `ss -tuln` (Linux) or Little Snitch (Mac) to audit connections.

Docker Needs Pre-Pulled Images

Docker containers can't pull images offline. If you use Docker-based tools (Open WebUI, vLLM), pull all images before disconnecting:

```
docker pull ghcr.io/open-webui/open-webui:main
docker pull vllm/vllm-openai:latest

# Verify images are cached
docker images
```

Docker runs these cached images without network access.

DNS and Localhost Resolution

Some systems need network-related services running even for localhost connections. If `localhost:11434` doesn't resolve offline:

```
# Ensure localhost is in /etc/hosts (Linux/Mac)
cat /etc/hosts | grep localhost
# Should show: 127.0.0.1 localhost
```

```
# Use 127.0.0.1 directly instead of localhost
curl http://127.0.0.1:11434/api/tags
```

Time/Date Drift

Systems offline for extended periods lose time synchronization. This usually doesn't affect AI inference, but can cause:

- TLS certificate errors if you reconnect briefly
- Incorrect timestamps in logs
- Issues with time-based file operations

If you reconnect periodically, time syncs automatically. For permanently air-gapped systems, consider setting up a local NTP source.

The Minimal Offline Stack

You don't need a complex setup. The absolute minimum:

```
Ollama + one model + a terminal = working offline AI
```

That's it. One binary, one model file, one command:

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

Works on any machine with 8GB RAM. No Python, no Docker, no web UI, no configuration files. Just a prompt and a response, running entirely on your hardware, with zero network dependency.

Everything else — RAG, image generation, web UIs, multiple models — is layered on top of this foundation. Start simple, add complexity only when you need it.

📚 **Setup guides:** First Local LLM · LM Studio Tips · AnythingLLM Setup

📚 **Hardware:** Laptop vs Desktop for AI · Mac M-Series Guide · VRAM Requirements

---

Source: https://insiderllm.com/guides/running-ai-offline-complete-guide/

Free guides for running AI locally