Managing Multiple Models in Ollama: Disk Space, Switching, and Organization

February 8, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Run `ollama list` to see every model and its size. Remove unused models with `ollama rm modelname:tag`. Check actual disk usage with `du -sh ~/.ollama/models/` (Linux/Mac) or check `C:\Users\<you>\.ollama\models` on Windows. Ollama auto-unloads models after 5 minutes idle — change this with `OLLAMA_KEEP_ALIVE=30m` for longer sessions. Use `ollama ps` to see what's currently loaded in VRAM. A typical collection of five 7B models takes ~20GB of disk; a 70B model alone takes ~40GB. Prune quarterly or whenever you're past 80% disk usage.

📚 **Related:** Ollama Setup Guide · Ollama Troubleshooting · Ollama vs LM Studio · VRAM Requirements · Planning Tool

You pulled a model to test. Then another. Then a 70B "just to see." Then a coding model, an embedding model, a couple of quantization variants. Three weeks later, your SSD is 100GB lighter and you're not sure what half of these models are.

This is the normal trajectory for anyone who uses Ollama regularly. Models are big, experimentation is easy, and Ollama doesn't warn you when your disk is filling up.

Here's how to see what you have, clean up what you don't need, and organize what you keep.

How Much Space Are Your Models Using?

Quick Check

```
ollama list
```

This shows every model, its tag, size, and when it was last modified:

```
NAME                    ID              SIZE      MODIFIED
qwen2.5:14b             a2e484b9a5ce    9.0 GB    2 days ago
qwen2.5:7b              845dbda0ea48    4.7 GB    3 days ago
```

```
llama3.1:8b              46e0c10c039e  4.9 GB   1 week ago
phi4:latest              7e510075d4a6  9.1 GB   2 weeks ago
nomic-embed-text:latest  0a109f422b47  274 MB   3 weeks ago
deepseek-coder-v2:16b    63fb193b3a9b  8.9 GB   1 month ago
llama3.1:70b-q4_K_M      etc           40.8 GB  2 months ago
```

Total Disk Usage

```
# Linux/Mac
du -sh ~/.ollama/models/

# Windows (PowerShell)
(Get-ChildItem -Path "$env:USERPROFILE\.ollama\models" -Recurse | Measure-Object -Property Lengt
```

A typical hobbyist collection lands between 30-80GB. Power users with 70B models and multiple variants can easily hit 200GB+.

Per-Model Breakdown

```
# Show each model with its size, sorted largest first
ollama list | tail -n +2 | sort -k3 -h -r
```

This sorts by size so you can spot the space hogs immediately. That 70B model you pulled once and forgot about? It's probably at the top.

How Ollama Stores Models

Understanding the storage structure helps when things go wrong or when you want to move models between machines.

Storage Location

OS	Default Path
Linux	`~/.ollama/models/`
macOS	`~/.ollama/models/`

OS	Default Path
Windows	`C:\Users\<you>\.ollama\models\`

To change the storage location, set the `OLLAMA_MODELS` environment variable before starting Ollama:

```
# Linux/Mac — move models to a larger drive
export OLLAMA_MODELS=/mnt/bigdrive/ollama-models
ollama serve
```

Useful if your SSD is small but you have a large HDD or secondary drive. Models load slower from HDD but this only affects the initial load — inference speed depends on GPU, not disk.

Inside the Directory

```
models/
├── blobs/          # Actual model weight files (large)
└── manifests/      # Metadata linking model names to blobs
    └── registry.ollama.ai/
        └── library/
            ├── qwen2.5/
            ├── llama3.1/
            └── ...
```

Blobs are the actual weights — files named by their SHA256 hash. These are what eat your disk space.

Manifests are small JSON files that map model names and tags to specific blobs. When you `ollama pull qwen2.5:7b`, the manifest records which blobs make up that model.

Shared layers: Models from the same family often share base blobs. For example, `qwen2.5:7b` and `qwen2.5:7b-instruct` may share weight layers with different adapter blobs on top. Deleting one doesn't necessarily free all its listed disk space — Ollama keeps shared blobs until all models referencing them are removed.

Removing Models

Single Model

```
ollama rm llama3.1:70b-q4_K_M
```

This removes the manifest and any blobs not shared with other models. If you have `llama3.1:8b` and `llama3.1:70b`, removing the 70B version won't affect the 8B.

Bulk Removal

Ollama doesn't have a bulk delete command, but you can script it:

```
# Remove all models matching a pattern
ollama list | grep "deepseek" | awk '{print $1}' | xargs -I {} ollama rm {}

# Remove all models (nuclear option - be sure!)
ollama list | tail -n +2 | awk '{print $1}' | xargs -I {} ollama rm {}
```

The "Dangling Blobs" Problem

Occasionally, blobs linger after model removal — especially if Ollama was interrupted during a pull or delete. To reclaim this space:

```
# Check for orphaned blobs (Linux/Mac)
du -sh ~/.ollama/models/blobs/

# Compare with total listed model sizes
ollama list
```

If the blobs directory is significantly larger than the sum of your listed models, there are orphans. The safest fix: remove all models with `ollama rm`, then re-pull the ones you want. Heavy-handed but guaranteed clean.

Switching Between Models

No “Switch” Needed

Ollama doesn’t have a mode switch. Every request specifies which model to use:

```
# Interactive chat — just name the model
ollama run qwen2.5:14b

# API calls — specify in the request
curl http://localhost:11434/api/generate -d '{
  "model": "qwen2.5:14b",
  "prompt": "Explain recursion"
}'
```

When you request a model that isn’t loaded, Ollama loads it into VRAM automatically. When a model sits idle, Ollama unloads it to free memory.

The Keep-Alive Timer

By default, Ollama unloads a model after **5 minutes** of inactivity. This is fine for occasional use but annoying if you’re switching between tasks — every time you come back after a coffee break, the model reloads (10-30 seconds depending on size and disk speed).

Change the timeout:

```
# Keep models loaded for 30 minutes
OLLAMA_KEEP_ALIVE=30m ollama serve

# Keep models loaded indefinitely (until manually unloaded or server restart)
OLLAMA_KEEP_ALIVE=-1 ollama serve

# Per-request override via API
curl http://localhost:11434/api/generate -d '{
  "model": "qwen2.5:14b",
  "prompt": "Hello",
  "keep_alive": "1h"
}'
```

Trade-off: Longer keep-alive means VRAM stays occupied even when you're not using the model. If you only have one GPU, a loaded model blocks other models from loading until it's evicted.

What's Currently Loaded?

```
ollama ps
```

Output:

```
NAME           ID              SIZE     PROCESSOR    UNTIL
qwen2.5:14b    a2e484b9a5ce    9.0 GB   100% GPU     4 minutes from now
```

This shows which model is in VRAM, how much space it's using, whether it's on GPU or CPU, and when it will auto-unload.

Running Multiple Models Simultaneously

If you have enough VRAM, Ollama can keep multiple models loaded at once.

VRAM Math

Each model needs its full allocation. There's no sharing or compression between loaded models:

Your VRAM	What Fits Simultaneously
8 GB	One 7B model at Q4 (~5GB)
12 GB	One 14B at Q4 (~9GB) OR two 7B models
16 GB	One 14B at Q4 + one embedding model
24 GB	Two 7B models + embedding model, or one 14B + one 7B

Embedding models like nomic-embed-text (~275MB loaded) are small enough to keep resident alongside a chat model on any setup.

Forcing Multiple Models Loaded

```
# Terminal 1: Load and keep a model
curl http://localhost:11434/api/generate -d '{
  "model": "qwen2.5:7b",
  "prompt": "test",
  "keep_alive": "-1"
}'

# Terminal 2: Load a second model
curl http://localhost:11434/api/generate -d '{
  "model": "nomic-embed-text",
  "prompt": "test",
  "keep_alive": "-1"
}'

# Verify both are loaded
ollama ps
```

If total VRAM demand exceeds your GPU capacity, the oldest model gets evicted to make room for the new one. Ollama handles this automatically — no crash, just a reload delay when you go back to the evicted model.

Manually Unloading

```
# Unload a specific model immediately
curl http://localhost:11434/api/generate -d '{
  "model": "qwen2.5:7b",
  "keep_alive": 0
}'
```

Useful when you're done with a model and want to free VRAM for something else without waiting for the timeout.

Organization Strategies

The Tiered Approach

Organize your models into three tiers:

Tier 1 — Daily drivers (keep always): Your go-to models for everyday use. For most people, this is one general chat model and one embedding model.

```
qwen2.5:14b          # General purpose
nomic-embed-text     # Embeddings for RAG
```

Tier 2 — Specialists (keep available): Models you use regularly for specific tasks. Keep them on disk, load when needed.

```
deepseek-coder-v2:16b    # Coding
phi4:latest              # Math/reasoning
```

Tier 3 — Experiments (delete after testing): Models you pulled to evaluate. Test them, decide if they earn a spot in Tier 1 or 2, then delete.

```
# After testing, remove what didn't make the cut
ollama rm mistral:7b
ollama rm gemma3:27b
```

Naming Custom Models

When you create custom models with Modelfiles (fine-tuned, custom system prompts, modified parameters), use a consistent naming scheme:

```
# Format: base-model:purpose-version
ollama create qwen2.5:14b-coding-v1 -f Modelfile.coding
ollama create qwen2.5:14b-writing-v2 -f Modelfile.writing
```

This makes cleanup easy — `ollama list | grep "coding"` shows all your coding variants.

Document What You Keep

Keep a simple text file alongside your models directory listing what you have and why:

```
# ~/ollama-models.txt
#
# Daily drivers:
# qwen2.5:14b — general purpose, best overall at this VRAM
# nomic-embed-text — embeddings for AnythingLLM
#
# Specialists:
# phi4:latest — math homework help, benchmark testing
# deepseek-coder-v2:16b — complex coding tasks
#
# Last cleanup: 2026-02-08
# Total disk: ~37 GB
```

It sounds excessive, but after your third time trying to remember why you pulled a specific model variant, you'll be glad you have it.

When to Clean Up

Before downloading a large model: If you're about to pull a 70B model (~40GB), make sure you have space. Check with `df -h` (Linux/Mac) or Task Manager → Disk (Windows).

When disk usage exceeds 80%: SSDs slow down when nearly full. Keep at least 20% free for system operations, swap space, and temporary files during model loading.

Quarterly habit: Set a calendar reminder. Pull up `ollama list`, sort by modification date, and ask yourself: "Have I used this in the last 3 months?" If not, remove it. You can always re-pull later.

After model upgrades: When Qwen 2.5 drops and you pull it, remove the Qwen 2 version you were running before. Old versions sitting around "just in case" are the #1 source of wasted disk space.

Quick Cleanup Checklist

```
# 1. See what you have
ollama list

# 2. Check total disk usage
du -sh ~/.ollama/models/

# 3. Remove models you don't use
```

```
ollama rm model:tag

# 4. Verify space was freed
du -sh ~/.ollama/models/

# 5. Check for orphaned blobs (optional)
# If disk usage seems high relative to listed models,
# consider removing all and re-pulling your keepers
```

Common Questions

“Can I copy models to another machine?”

Yes. Copy the entire `~/.ollama/models/` directory to another machine with Ollama installed. The blobs and manifests are self-contained. This is faster than re-downloading over a slow connection.

```
# From source machine
rsync -av ~/.ollama/models/ user@other-machine:~/.ollama/models/
```

“Can I use an external drive for model storage?”

Yes. Set `OLLAMA_MODELS` to point to any mounted drive. Models will load slower from HDD or USB, but inference speed is unaffected once loaded into VRAM.

“How do I update a model to the latest version?”

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

This downloads only the changed layers — if the model was updated but shares base weights with your existing version, the download is smaller than a fresh pull. The old version is replaced automatically.

For more on updates, see the Ollama troubleshooting guide.

"Do quantization variants share disk space?"

No. `qwen2.5:14b` (Q4 default) and `qwen2.5:14b-q8_0` are separate downloads with separate blobs. Each variant takes its full disk space. Don't keep multiple quantizations of the same model unless you have a specific reason.

The Bottom Line

Model management in Ollama is simple once you build the habit: `ollama list` to see what you have, `ollama rm` to clean up, `ollama ps` to check what's loaded. The disk space problem is real — five models accumulate 30GB+ before you notice — but it's a five-minute fix once you know the commands.

The most important habit: delete experiments after testing. Every model you pull "just to try" should earn its spot on disk or get removed. Future you will thank present you when the SSD isn't mysteriously full.

📚 **Ollama guides:** First Local LLM Setup · Ollama Troubleshooting · Ollama vs LM Studio · llama.cpp vs Ollama vs vLLM

📚 **Model choices:** VRAM Requirements · Best Models for Chat · Qwen Guide

---

Source: https://insiderllm.com/guides/managing-multiple-models-ollama/

Free guides for running AI locally