How to Update Models in Ollama — Keep Your Local LLMs Current

February 14, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Ollama never auto-updates models. To update one, run `ollama pull model:tag` — it re-downloads only changed layers, so it's fast if the update is small. To update everything at once: `ollama list | tail -n +2 | awk '{print $1}' | while read model; do ollama pull "$model"; done`. Check what you have with `ollama list` (the MODIFIED column shows when you last pulled). Use `ollama show model:tag` to see the exact architecture, quantization, and parameters. If a model is working well for your use case, there's no rush to update — new versions occasionally change behavior.

📚 **Related:** Managing Multiple Models · Ollama Troubleshooting · Ollama vs LM Studio · Run Your First Local LLM

Qwen pushes a fix. Llama drops a point release. DeepSeek patches a tokenizer issue. These updates happen constantly, but Ollama doesn't tell you about any of them. There's no notification, no auto-update, no "new version available" banner. Your models stay exactly where you left them until you explicitly pull again.

That's actually fine — you don't want models silently changing under you. But it means you need a system for checking what's stale and deciding when to update.

How Ollama Updates Work

There's no `ollama update` command. The update mechanism is just `ollama pull` — the same command you used to download the model in the first place.

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

When you run this on a model you already have, Ollama checks the registry for the latest version of that tag. Three things can happen:

1. **No changes** — Ollama verifies each layer, prints "already exists" for everything, and finishes in seconds. Nothing downloaded.

2. **Partial update** — Some layers changed (bug fix, fine-tuning adjustment). Ollama downloads only the changed layers and reuses the rest. A minor update to a 9GB model might only download 200MB.
3. **Full re-download** — The model was rebuilt from scratch (new base weights, different architecture). Everything downloads fresh.

In practice, most updates are case 1 or 2. Full re-downloads are rare and usually happen when a model family releases a new major version (like Llama 3.1 → Llama 3.2).

Tags Are Mutable

Here's the part that trips people up: **tags aren't version numbers**. When you pull `qwen2.5:7b`, you're pulling whatever the Ollama library currently points that tag at. If Qwen's team pushes an updated quantization next week, `qwen2.5:7b` now refers to the new version. Your local copy stays the same until you pull again.

This is why `ollama list` shows a "MODIFIED" date — that's when you last pulled, not when the model was created.

Check What You Have

See All Models

```
ollama list
```

```
NAME                    ID            SIZE    MODIFIED
qwen2.5:14b             a2e484b9a5ce  9.0 GB  3 weeks ago
llama3.1:8b             46e0c10c039e  4.9 GB  2 months ago
deepseek-coder-v2:16b   63fb193b3a9b  8.9 GB  2 months ago
phi4:latest             7e510075d4a6  9.1 GB  1 month ago
```

That MODIFIED column is your staleness indicator. Anything older than a month is worth checking. Anything older than three months has almost certainly been updated upstream.

Get Detailed Model Info

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

This prints the model's architecture, parameter count, quantization level, context length, and system prompt template:

```
  Model
    architecture        qwen2
    parameters          14.8B
    quantization        Q4_K_M
    context length      32768
    embedding length    5120

  Parameters
    stop    "<|im_end|>"
    stop    "<|endoftext|>"

  License
    Apache License 2.0
```

Use this to verify you're running the quantization you think you're running. It's also helpful when comparing pre- and post-update to see if anything changed.

Check a Specific Model's Modelfile

```
ollama show --modelfile qwen2.5:14b
```

This shows the full Modelfile including the base layer digest, parameters, template, and system prompt. The SHA256 digest at the top is the fingerprint — if this changes after a pull, the model weights changed.

Updating a Single Model

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

Watch the output. You'll see one of two patterns:

Already current (no update):

```
pulling manifest
pulling a]... 100% ████████████████  already exists
pulling b]... 100% ████████████████  already exists
verifying sha256 digest
writing manifest
success
```

Update available (new layers downloading):

```
pulling manifest
pulling a]... 100% ████████████████  already exists
pulling c]...  47% ████████        │  2.1 GB/4.5 GB
```

When new layers download, the old ones are replaced. There's no manual cleanup needed — Ollama handles the swap.

Verify the Update

After pulling, check that the model info changed:

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

Compare the output with what you saw before. Changes in the parameters, quantization level, or template indicate a meaningful update. If everything looks identical, the update was either a metadata-only change or there was no update at all.

Updating All Models at Once

Ollama doesn’t have a built-in “update all” command, but it’s one line:

```
ollama list | tail -n +2 | awk '{print $1}' | while read model; do
  echo "Updating $model..."
  ollama pull "$model"
  echo ""
done
```

This loops through every model in `ollama list`, skips the header row, and pulls each one. Models already current finish in seconds. Models with updates download the new layers.

A Better Version

The basic loop works, but it doesn’t tell you what actually changed. This version tracks updates:

```
#!/bin/bash
echo "Checking for model updates..."
echo ""

ollama list | tail -n +2 | awk '{print $1}' | while read model; do
  old_id=$(ollama list | grep "^$model " | awk '{print $2}')
  ollama pull "$model" > /dev/null 2>&1
  new_id=$(ollama list | grep "^$model " | awk '{print $2}')

  if [ "$old_id" != "$new_id" ]; then
    echo "UPDATED: $model ($old_id → $new_id)"
  else
    echo "OK: $model (already current)"
  fi
done
```

Save this as `update-ollama-models.sh`, make it executable with `chmod +x update-ollama-models.sh`, and run it whenever you want to check for updates.

When to Update vs When to Stay Put

Not every update is worth grabbing. Here's how to think about it.

Update when:

- **You're hitting bugs.** Garbled output, wrong stop tokens, or template issues are often fixed in point releases.
- **A new quantization is available.** Model maintainers sometimes re-quantize with better methods, giving you better quality at the same size.
- **You're starting a new project.** Fresh pull ensures you're on the latest version before building something that depends on specific model behavior.
- **It's been 3+ months.** Major model families (Qwen, Llama, Mistral) push meaningful improvements regularly.

Stay put when:

- **Your current setup works well.** If you've tuned prompts and parameters around a specific model version, updating can break that. New versions sometimes change the system prompt template or default parameters.
- **You're mid-project.** Updating a model you're actively using for consistent output (like generating training data or writing structured content) can introduce subtle changes you don't notice immediately.
- **Disk space is tight.** Even delta updates temporarily need space for both old and new layers during the swap.

The Breaking Change Problem

The most common breaking change is **template format**. When a model family updates their chat template (the `<|im_start|>` / `<|im_end|>` structure, or `[INST]` markers), tools that send raw prompts — not just `ollama run` but API integrations, Open WebUI, custom scripts — can break.

Before updating a model you've integrated into a workflow, check the model's page on ollama.com/library for release notes or changelogs. If the template changed, you'll need to update your integration too.

Cleaning Up After Updates

When you `ollama pull` an updated model, the new layers replace the old ones. You don't end up with two copies of the same model. But there are edge cases.

Different Tags Are Different Models

`qwen2.5:7b` and `qwen2.5:14b` are separate models. Updating one doesn't affect the other. If you pulled `qwen2.5:7b` to test and then moved to `qwen2.5:14b` permanently, the old 7B is still sitting on disk:

```
# See what's taking space
ollama list

# Remove models you no longer use
ollama rm qwen2.5:7b
```

Shared Layers

Models from the same family sometimes share base layers. For example, `qwen2.5:7b` and `qwen2.5:7b-instruct` might share weight blobs with different adapter layers on top. Removing one doesn't necessarily free all the listed disk space — Ollama keeps shared blobs until every model referencing them is removed.

Check Disk Usage

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

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

If this number seems high compared to the total sizes in `ollama list`, you might have orphaned blobs from interrupted pulls or updates. The nuclear option: remove all models with `ollama rm`, then re-pull your keepers. For a deeper dive on this, see the managing multiple models guide.

Automating Updates with Cron

If you want weekly update checks without thinking about it, add a cron job:

```
crontab -e
```

Add this line to check every Sunday at 3 AM:

```
0 3 * * 0 ollama list | tail -n +2 | awk '{print $1}' | while read model; do ollama pull "$model"
```

Check `/tmp/ollama-updates.log` to see what happened.

Why You Might Not Want This

Automated updates are convenient but risky if you're running models in production-like setups (API servers, scheduled tasks, integrations). A silent template change at 3 AM can break your Monday morning workflow.

A safer approach: automate the check but not the pull. Run `ollama pull --dry-run` — except Ollama doesn't support dry runs yet. So the practical alternative is the ID-comparison script from earlier: run it on a schedule, log which models have updates available, and pull manually when you're ready.

The Bottom Line

`ollama pull model:tag` is both the install and the update command. Run it periodically on models you care about, skip it when things are working. Use the update-all script for maintenance days, and don't automate pulls on models you've integrated into workflows without checking for breaking changes first.

📚 **Ollama guides:** Managing Multiple Models · Ollama Troubleshooting · Ollama vs LM Studio · llama.cpp vs Ollama vs vLLM

📚 **Model choices:** Qwen Guide · DeepSeek Guide · Llama 3 Guide · VRAM Requirements

---

Source: https://insiderllm.com/guides/update-models-ollama/

InsiderLLM.com

Free guides for running AI locally