

Install Ollama, **run Olmo 3**

1 Install

macOS and Windows: download the app from ollama.com/download. Linux:

```
curl -fsSL \
  https://ollama.com/install.sh | sh
```

2 Run

The first run downloads the model, then you can chat. Type /bye to leave.

```
ollama run olmo-3:7b-instruct
```

3 Check

Is the model on your GPU, your CPU, or split between them?

```
ollama ps
# PROCESSOR: 100% GPU
```

OLMO 3 VARIANT	COMMAND	DOWNLOAD	GOOD FOR
Olmo 3 7B Instruct	<code>ollama run olmo-3:7b-instruct</code>	About 4.5 GB	Direct answers and chat
Olmo 3 7B Think	<code>ollama run olmo-3:7b-think</code>	About 4.5 GB	Step-by-step reasoning
Olmo 3 32B Think	<code>ollama run olmo-3:32b-think</code>	About 19 GB	Hardest problems, needs 24 GB or more

Use it from code, **tune it**

```
curl http://localhost:11434/api/chat -d '{
  "model": "olmo-3:7b-instruct",
  "messages": [{"role": "user",
                  "content": "Hello!"}]
}'
# Python: pip install ollama
from ollama import chat
r = chat(model='olmo-3:7b-instruct',
         messages=[{'role': 'user', 'content': 'Hi'}])
print(r.message.content)
```

Ollama also offers an OpenAI-compatible endpoint at `localhost:11434/v1`, so many existing tools work unchanged.

`OLLAMA_CONTEXT_LENGTH=8192`

Default context is 4096 tokens. More context costs more memory.

`ollama stop olmo-3:7b-instruct`

Models stay loaded for 5 minutes by default. Stop one to free memory now.

`OLLAMA_NO_CLOUD=1`

Local-only mode: turns off Ollama's cloud models and web search.

`127.0.0.1:11434`

Ollama listens on your machine only, unless you change `OLLAMA_HOST`.

Compare **your options**

Hardware

Apple Silicon

One pool of unified memory, so big models fit. Speed follows memory bandwidth.

GPU (NVIDIA, AMD)

Fastest when the model fits in video memory. Spills to slower system RAM when it does not.

CPU only

Works almost anywhere. Slower, so best for small models.

Runtimes

Ollama

Easy command line and API, with a model library. MIT license.

llama.cpp

The engine under many tools. Most control. MIT license.

LM Studio

Desktop app with a model browser. Free, but not open source.

vLLM

Serving many users on GPUs. Apache 2.0.

Model families

Fully open

Olmo: weights, training data and code are public.

Open weights, permissive

Qwen, gpt-oss, Phi, DeepSeek and many Mistral models, usually Apache 2.0 or MIT.

Open weights, custom terms

Llama and Gemma, with their own licenses.

Names and versions change. Read each model card.

Rule of thumb: pick the largest model whose weights fit with headroom, then run `ollama ps` to see where it landed.

Choosing and running **any** local model

- ☐ How much memory can the model really use?
- ☐ Do the weights fit? $\text{Parameters} \times \text{bits} \div 8$
- ☐ How much context do I need, and what does it cost?
- ☐ Dense or mixture-of-experts: memory versus speed?
- ☐ Is the license fully open, open weights, or restricted?
- ☐ Does my runtime support this model's architecture?
- ☐ Is anything leaving the machine? Cloud, network, telemetry
- ☐ How will I check the answers? Models make mistakes

DISCUSSION

Which of your tasks could use a model that never leaves your device, and which still need a bigger hosted one?

What do you gain when a model's training data and code are public too, not only its weights?