

INSTRUCTOR GUIDE AND NOTES

Run AI on **your own** computer

Local language models with Ollama and Olmo 3. A 50-minute lesson on why to run a model locally, how to tell what fits, and how to install and use one.

Series	Hack Your Future, PrivacySafe Foundation's series for building what's next
Audience	Learners comfortable opening a terminal. No machine-learning background needed.
Length	50 minutes, with a 30-minute cut
Format	4 slides, live planner demo, 4 slides, hands-on lab, discussion
You will need	Slides (light for print, dark for projecting), the model planner page, flash cards, a computer with Ollama installed
Facts checked	October 27, 2026. Model names, sizes and defaults change. Re-check before you teach.

PrivacySafe Foundation, Inc. | Hack Your Future | These materials are dedicated to the public domain under CC0 1.0 and shared for the public good. The PrivacySafe and PrivacySafe Foundation names and logos are trademarks and are not included in that dedication. Ollama, Olmo and other third-party names belong to their owners.

The lesson in five sentences

1. A local model is three things working together: a weights file, a runtime, and your hardware. The file for Olmo 3 7B is about 4.5 GB and for 32B about 19 GB^[4].
2. Memory is the limit: weights, plus the context cache, plus overhead must fit. Weights are roughly parameters times bits per weight divided by 8.
3. Memory bandwidth, more than raw compute, sets how fast words appear, because every token reads the weights.
4. Ollama installs in minutes, runs a model with one command, and shows where it loaded with `ollama ps`^{[3][1]}.
5. Local is not automatically private. Check cloud features, network binding and where the model came from^[1].

LEARNING OUTCOMES

By the end, students can:

1. Explain why someone might run a model locally, and the honest trade-offs.
2. Estimate whether a model fits a given machine using parameters, quantization and context.
3. Install Ollama, run an Olmo 3 model, and read `ollama ps`.
4. Call a local model from the command line, from code, or through an OpenAI-compatible client^[9].
5. Compare hardware, runtimes and model families using questions that stay useful as models change.

Session plan

SEGMENT	MINUTES	NOTES
Before the demo, slides 1 to 4	12	Includes the prediction. Collect answers silently.
Planner demo	8	Five presets. The 16 GB laptop presets answer the prediction.
After the demo, slides 1 to 4	10	Install, use, compare, checklist.
Hands-on lab	12	Install and run a model. See the lab page. Pre-download if bandwidth is limited.
Discussion	8	Two questions. Pairs first, then the room.

30-MINUTE CUT

Skip the hands-on lab and show it on the instructor machine instead. Run only the first three planner presets. Use one discussion question.

Before class

- Optional: the lesson page has a walkthrough video of under five minutes, in a light version for the page and a dark version for projecting, each with a subtitles file. Play it as a recap, or share it with learners who missed the session. Check that it plays on the room computer first.
- **Install Ollama and pull a model beforehand.** Run `ollama run olmo-3:7b-instruct` once on the instructor machine so the 4.5 GB download is done^[4].
- **Plan the downloads.** If 20 learners pull 4.5 GB at once on shared Wi-Fi, the room will crawl. Ask learners to download the night before, bring a USB copy, or watch the instructor demo.
- Check learners' machines: free disk space (several GB per model), memory, and whether they have a supported GPU^[2].
- Open the planner page and run each preset once. Set browser zoom to about 125% for the back row.
- Run `ollama ps` on your machine and note the PROCESSOR column. You will refer to it.
- Decide on privacy settings for the room: set `OLLAMA_NO_CLOUD=1` on the instructor machine^[1].
- Print the flash cards, double-sided, flipped on the long edge. Cut along the dashed lines.

WHAT THE PLANNER IS, AND IS NOT

The planner runs in the browser and sends none of your choices anywhere. It estimates memory with the formulas on the slides and estimates speed as roughly 60% of memory bandwidth divided by the bytes read per token. These are teaching estimates, not benchmarks. The Olmo 3 layer, head and context numbers come from the model cards^[7]. Olmo 3 mixes sliding-window and full-attention layers, so its real cache use at long context can be lower than the planner shows. Bandwidth classes are typical vendor figures. Always confirm with `ollama ps` on the real machine^[1].

Slide notes: before the demo

Slide 1: Title

about 2 min

SAY

Can a capable language model run on hardware you already own? The promise: your prompts stay on your machine and you choose the model. Our job is to pick a model that fits, install Ollama, and run Olmo 3.

DO

Point at the memory bar at the bottom. It returns on slide 3 and in the planner.

CHECK FOR UNDERSTANDING

Ask: "Who has used an AI chatbot? Where do you think your question went?"

Slide 2: Why run a model locally?

about 4 min

SAY

	HOSTED	LOCAL
Where prompts go	A provider's servers	Stay on your machine
Cost	Per token or subscription	Hardware and power you already have
Offline	Needs a connection	Works after the download
Control	Provider can change or retire the model	You pick the version and keep the files
Capability	Largest models, fast	Smaller models, limited by memory

End on the honest trade-off and the warning: local is not automatically private.

CHECK FOR UNDERSTANDING

Ask learners to name one task where local is clearly better and one where hosted is. There is no single right answer.

WATCH FOR

Do not oversell. Smaller local models are less capable than the largest hosted ones. The warning matters: cloud features, network binding and model provenance all affect privacy^[1].

Slide 3: What is inside a local model?

about 4 min

SAY

- Layer 1, **weights**: billions of numbers in one file.
- Layer 2, **runtime**: the program that loads weights and does the math. Ollama builds on engines such as llama.cpp^{[8][3]}.

- Layer 3, **hardware**: memory holds the model, and bandwidth sets speed.

Memory is about parameters times bits per weight divided by 8, plus context and overhead.

Quantization stores each weight in fewer bits. Going from 16-bit to 4-bit makes the file about four times smaller, with a small quality cost.

DO

Work the example on the board: 7.3 billion weights at about 4.85 bits is about 4.4 GB. The Ollama page lists about 4.5 GB for Olmo 3 7B^[4]. The 32B model comes to about 19 GB.

CHECK FOR UNDERSTANDING

Ask: "If I halve the bits, what happens to the file?" It roughly halves.

WATCH FOR

The 4.85 figure is an average for the common 4-bit format. It is above 4 because some tensors are stored with more bits and each block carries scaling values. Learners may ask why not exactly 4.

Slide 4: What to watch for, and the prediction

about 2 min

SAY

Watch for: memory is the limit, bandwidth sets speed, context costs memory, dense versus mixture-of-experts, and 4-bit as the common default. On the Ollama side: `run`, `pull`, `ps`, `stop`, the Olmo 3 sizes, Instruct versus Think, and cloud features.

DO

Read the prediction aloud: "Can a laptop with 16 GB of memory run Olmo 3 32B Think? What about 7B Instruct?" Give 60 seconds of silence. Take a show of hands for "both", "only 7B", "neither", "only 32B".

WATCH FOR

Many learners expect a 32B model to "just run slowly" on 16 GB. Slow is only possible if the model fits somewhere.

Planner demo script (about 8 minutes)

Preset 1: Olmo 3 7B on a 16 GB laptop

1 min

SAY

The planner shows about 7.6 GB needed: 4.4 GB of weights, 2.1 GB of context cache at the default 4096 tokens, and about 1 GB of overhead. It fits in the memory the GPU can use, at roughly 16 tokens per second on a base Apple Silicon chip.

DO

Click the preset. Point at the stacked bars: weights, context cache, overhead against the memory you have. Show the "Commands for this setup" card and the sample `ollama ps` output.

CHECK FOR UNDERSTANDING

Ask: "Why is the context cache so large for a 7B model?" Olmo 3 7B has 32 key-value heads and no grouped-query attention^[7], so each token costs more cache than in many newer models.

Preset 2: Olmo 3 32B Think on a 16 GB laptop

1 min

SAY

About 21.5 GB needed against roughly 11 GB the GPU can use, and about 12 GB of total usable memory. It does not fit. Close the prediction: 7B yes, 32B no.

DO

Click the preset. The verdict turns red and suggests a fix, such as Olmo 3 7B Instruct.

WATCH FOR

Learners may try "swap". Using disk as memory makes a model effectively unusable. The planner reports it as too big.

Preset 3: Olmo 3 32B Think on a 24 GB GPU

1 min

SAY

About 21.5 GB needed against about 23.5 GB of video memory. It fits, but tight: little room to spare, around 30 tokens per second on a high-end GPU.

DO

Click the preset. Then raise the context window to 16,384 and watch it tip into a split between GPU and system RAM. Explain the PROCESSOR column^[1].

CHECK FOR UNDERSTANDING

Ask: "What are two ways to make it fit more comfortably?" A shorter context, the 8-bit context cache, or a smaller model.

Preset 4: an 8B-class model at 128K context

2 min

SAY

Context is not free. At 128K tokens the cache alone is about 17 GB, more than the weights. Long context needs memory for the cache as well as the model.

DO

Click the preset. Show that the weights are about 4.9 GB while the context cache dominates.

CHECK FOR UNDERSTANDING

Ask: "Why does Ollama default to a modest context?" Memory. The default is 4096 tokens^[1]. The Olmo 3 models support up to 65,536^[7].

Preset 5: a 70B-class model on 64 GB

2 min

SAY

A 70B dense model at 4-bit needs about 45 GB. It fits a 64 GB Apple Silicon machine with a Max-class chip, but speed is only a few tokens per second.

DO

Switch the size class to the mixture-of-experts options and compare. A model with 30B total and 3B active parameters needs memory for all 30B but reads only about 3B per token, so it can be much faster.

CHECK FOR UNDERSTANDING

Ask: "Which number sets memory, and which sets speed, for a mixture-of-experts model?" Total parameters set memory. Active parameters set speed.

Close the prediction

1 min

SAY

Return to slide 4. 7B Instruct fits a 16 GB laptop. 32B Think does not.

DO

Ask who guessed correctly. Remind learners the planner is an estimate: the check on a real machine is `ollama ps`.

Slide notes: after the demo

Slide 1: Install Ollama, run Olmo 3

about 3 min

SAY

Install: on macOS and Windows, download the app from ollama.com/download. On Linux, use the install script^{[1][3]}. Run: `ollama run olmo-3:7b-instruct`. The first run downloads the model, then you can chat. Check: `ollama ps` shows whether it is on the GPU, the CPU, or split.

DO

Walk through the variants table. Instruct answers directly. Think shows step-by-step reasoning. The 32B Think download is about 19 GB^[4].

CHECK FOR UNDERSTANDING

Ask: "If PROCESSOR says 100% CPU, what does that tell you?" The model is running in system memory and will be slower.

WATCH FOR

Olmo 3 on Ollama: use the exact tags shown on the library page^[4]. Olmo 3.1 and Olmo 2 are different releases and are not the subject of this lesson.

Slide 2: Use it from code, tune it

about 3 min

SAY

A local server listens on 127.0.0.1 port 11434 by default^[1]. You can call `/api/chat`, use the Python or JavaScript libraries, or point an OpenAI-style client at `/v1`^[9]. Settings that matter:

`OLLAMA_CONTEXT_LENGTH` (default 4096), `ollama stop` (models stay loaded for 5 minutes), and `OLLAMA_NO_CLOUD=1` for local-only mode^[1].

WATCH FOR

Setting `OLLAMA_HOST` to 0.0.0.0 exposes the server to your network^[1]. Do not do that on a shared Wi-Fi network.

Slide 3: Compare your options

about 4 min

SAY

- **Hardware:** Apple Silicon has one pool of unified memory. GPUs are fastest when the model fits in video memory and spill to slower system RAM when not. CPU-only works almost anywhere but is slower^[2].
- **Runtimes:** Ollama is an easy command line and API^[3]. llama.cpp is the engine under many tools^[8]. LM Studio is a desktop app that is free but not open source^[12]. vLLM serves many users^[11].
- **Model families:** fully open, open weights with permissive licenses, and open weights with custom terms^{[13][14][15]}.

DO

Keep this about questions, not product names. Names and versions change. The questions to ask stay the same: what license, what size, what architecture, which runtime supports it.

CHECK FOR UNDERSTANDING

Ask: "What is the difference between open weights and fully open?" Open weights give you the parameters. Fully open also releases the code and detailed data information^[13]. Ai2 describes releasing training code, data, weights and recipes^[5].

WATCH FOR

Licenses change from release to release. Tell learners to read the model card. Do not teach a license from memory.

Slide 4: Choosing and running any local model

about 4 min

SAY

1. How much memory can the model really use?
2. Do the weights fit? Parameters times bits divided by 8.
3. How much context do I need, and what does it cost?
4. Dense or mixture-of-experts: memory versus speed?
5. Is the license fully open, open weights, or restricted?
6. Does my runtime support this architecture?
7. Is anything leaving the machine?
8. How will I check the answers? Models make mistakes.

DO

Optional activity: pick any model from the Ollama library and have pairs answer the eight questions from its page.

Hands-on lab (about 12 minutes)

Goal: every learner runs a model and reads `ollama ps`. Use the instructor machine for anyone whose laptop cannot run a 7B model.

1. **Install.** Download from ollama.com/download (macOS, Windows) or run the Linux install script^[3]. Open a terminal and run `ollama --version`.
2. **Run.** `ollama run olmo-3:7b-instruct`. Wait for the download. Ask: "Explain what a language model is in two sentences." Type `/bye` to leave.
3. **Check.** In a second terminal run `ollama ps`. Note SIZE and PROCESSOR. Run `ollama list` to see what is on disk.
4. **Compare.** Run `ollama run olmo-3:7b-think` with the same prompt. Notice the reasoning before the answer and the longer wait^[4].
5. **Change the context.** Inside `ollama run`, type `/set parameter num_ctx 8192`, then run `ollama ps` again and compare SIZE^[1].
6. **Use the API.** `curl http://localhost:11434/api/chat` with a JSON body naming the model and a user message.
7. **Go local-only.** Set `OLLAMA_NO_CLOUD=1`, restart Ollama, then turn off the network and ask another question. It still answers^[1].
8. **Free memory.** `ollama stop olmo-3:7b-instruct`^[1].

```
curl http://localhost:11434/api/chat -d '{
  "model": "olmo-3:7b-instruct",
  "messages": [{"role": "user", "content": "Hello!"}]
}'
```

Troubleshooting

SYMPTOM	LIKELY CAUSE AND FIX
Very slow replies	Check <code>ollama ps</code> . 100% CPU means no GPU is in use. Check GPU support and drivers ^[2] , or choose a smaller model.
Model will not load or the system freezes	The model plus context is too big. Use a smaller model, 4-bit quantization or a shorter context. Use the planner to check.
Download stalls or fails	Re-run the same command. Check free disk space. Models are stored in a <code>.ollama/models</code> folder whose location depends on the operating system ^[1] .
Model answers badly	Small models are limited. Try the Think variant for reasoning, a larger model if it fits, or a clearer prompt. Always verify important answers.
Port 11434 already in use	Ollama may already be running as a service or app. Use that instance.
GPU not detected on Linux after sleep	A known driver issue after suspend and resume is described in the Ollama GPU docs ^[2] .

Discussion (about 8 minutes)

QUESTION 1: WHICH TASKS STAY ON YOUR DEVICE?

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

- Good local candidates: private notes, drafting, summarizing documents you cannot upload, offline use, learning how models work.
- Hosted may still win for the hardest reasoning, very long documents, and the largest models.
- Ask learners to name what they would check before trusting either: provider terms for hosted, cloud settings and network exposure for local^[1].

QUESTION 2: WEIGHTS ONLY, OR THE WHOLE RECIPE?

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

- Open weights let you run and fine-tune. Fully open releases also let others inspect, reproduce and audit^{[13][5]}.
- Olmo 3 comes from Ai2 with weights, code and data information, and is licensed Apache 2.0^[7].
- Tension to explore: some data cannot be redistributed. The Open Source AI Definition asks for detailed data information, not always the raw data^[13].

Answer key and assessment

QUESTION	ANSWER
Roughly how much memory do the weights of a 7B model need at 4-bit?	About 4.4 GB, close to the 4.5 GB download ^[4] .
Which three things add up to the memory a model needs?	Weights, the context cache, and overhead.
What does the PROCESSOR column show?	Whether the model is on the GPU, the CPU, or split ^[1] .
What is the default context length in Ollama?	4096 tokens ^[1] .
Which setting turns off Ollama's cloud features?	<code>OLLAMA_NO_CLOUD=1</code> ^[1] .
Will Olmo 3 32B Think run on a 16 GB laptop?	No. About 21.5 GB is needed at 4096 tokens of context. 7B Instruct fits.

EXIT TICKET (3 MINUTES)

1. Write the memory formula for the weights.
2. Name two things that make a local setup less private than it looks.
3. What command tells you whether a model is on the GPU or the CPU?

HOMEWORK

- Pick a model from the Ollama library and use the planner to decide whether it fits your machine. Report the memory estimate and what `ollama ps` actually showed.
- Compare the Instruct and Think variants of Olmo 3 on three prompts. Describe the differences in answer quality and wait time.
- Stretch: write a short script that calls the local API and prints the reply.

Common questions and misconceptions

QUESTION OR BELIEF	RESPONSE
"Local means private."	Mostly, but not automatically. Ollama's cloud models and web search send data to a service, and exposing the port exposes the server ^[1] . Turn cloud features off if you want local-only.
"A bigger context window is always better."	It costs memory and can slow things down. Use what you need.
"4-bit models are broken."	4-bit is the common default. Quality loss is usually small but can matter for some tasks. Try 8-bit if it fits.
"More parameters always means better answers."	Usually, but training, data and tuning matter. Size sets what fits.
"Open weights and open source are the same."	No. Open source for AI asks for the code and data information as well ^[13] .
"My GPU is supported, so everything fits."	Support is about drivers. Fit is about memory ^[2] .

Choosing a model: what to check on a model card

CHECK	WHY
Parameters and architecture (dense or mixture-of-experts)	Sets memory and speed.
Quantization and download size	Sets whether it fits.

Context length	Sets cache cost and what documents you can use.
License and terms	Fully open, permissive open weights, or custom terms ^[13] ^{[14][15]} .
Runtime support	The architecture must be supported by your runtime ^[10] ^[8] .

References and further reading

Citations in this guide use bracketed numbers. Links were checked on October 27, 2026. Local AI tools change quickly, so confirm commands, defaults and model sizes against the current pages.

#	SOURCE	WHAT IT SUPPORTS
[1]	Ollama documentation, FAQ https://docs.ollama.com/faq	Default 4096-token context; ollama ps and the PROCESSOR column; keep-alive of 5 minutes; OLLAMA_HOST binding to 127.0.0.1:11434; OLLAMA_NO_CLOUD; KV cache types; model storage; updating.
[2]	Ollama documentation, Hardware support https://docs.ollama.com/gpu	NVIDIA, AMD (ROCm), Apple (Metal) and experimental Vulkan support.
[3]	Ollama on GitHub https://github.com/ollama/ollama	The project, the MIT license, and the Linux install script.
[4]	Ollama library, olmo-3 https://ollama.com/library/olmo-3	Olmo 3 tags and commands, download sizes (about 4.5 GB for 7B and 19 GB for 32B), 64K context, Instruct and Think variants.
[5]	Ai2 documentation, Introduction https://docs.allenai.org/	Ai2's description of Olmo 3 and of releasing training code, data, weights and recipes.
[6]	Ai2, Olmo 3 technical report page https://allenai.org/papers/olmo3	The Olmo 3 paper, referenced from the model cards.
[7]	Ai2 on Hugging Face (Olmo 3 model cards) https://huggingface.co/allenai	Model sizes, architecture table, 65,536-token context, Dolma 3 and Dolci data, Apache 2.0 license.
[8]	llama.cpp on GitHub https://github.com/ggml-org/llama.cpp	The C++ inference engine and the GGUF format that many local tools build on.
[9]	Ollama documentation, OpenAI compatibility https://docs.ollama.com/api/openai-compatibility	The /v1 endpoint that lets OpenAI-style clients talk to a local Ollama server.
[10]	Ollama documentation, Importing models https://docs.ollama.com/import	Model formats Ollama can import (GGUF and Safetensors).

[11]	vLLM documentation https://docs.vllm.ai/	A serving system for running models for many users.
[12]	LM Studio https://lmstudio.ai/	A desktop application with a model browser. Free to use; not open source.
[13]	Open Source Initiative, The Open Source AI Definition https://opensource.org/ai/open-source-ai-definition	What open source means for AI: data information, code and parameters, plus the four freedoms.
[14]	Meta, Llama license https://www.llama.com/llama3/license/	An example of open weights under a custom license.
[15]	Google, Gemma terms of use https://ai.google.dev/gemma/terms	An example of open weights under custom terms.
[16]	Creative Commons, CC0 1.0 Universal https://creativecommons.org/publicdomain/zero/1.0/	The public-domain dedication that covers these lesson materials.

ABOUT THESE MATERIALS

The slides, planner, flash cards and this guide were written for the PrivacySafe Foundation Hack Your Future series. They are dedicated to the public domain under CC0 1.0^[16]. The Foundation names and logos are trademarks and are not included in that dedication. Ollama, Olmo and other product names belong to their owners, and this lesson is independent of them.