PROJECT LUNA · FREE SAMPLE

Private local AI setup guide

A short, practical guide to running a language model on a Mac with Apple Silicon using llama.cpp and llama-server, including how to think about GGUF quantizations and RAM, how to enable vision with an mmproj file, and how to keep the server private.

November 2026 · Educational technical sample · Not legal advice · This guide was not tested on your computer.

What this is. A free sample from Project Luna. It walks through official install and run options published by the llama.cpp project. Commands and flags below are taken from those docs, as fetched on 17 November 2026 (America/New_York).

What this is not. It is not a review of any commercial model, not a claim that a particular model will fit a particular Mac, and not a guarantee that every vision model works with every llama.cpp build. Software changes. If a flag in this PDF disagrees with the live documentation, trust the live documentation.

1. Official sources

Use these as the source of truth:

- Project repository: https://github.com/ggml-org/llama.cpp (MIT license)
- Install and overview: the repository README, including https://llama.app and the releases page
- Build guide (macOS / Metal): docs/build.md
- Server (llama-server) usage and flags: tools/server/README.md
- Multimodal / vision: docs/multimodal.md
- Pre-built macOS Apple Silicon binaries: https://github.com/ggml-org/llama.cpp/releases

The older repository path `ggerganov/llama.cpp` redirected into this official `ggml-org/llama.cpp` project. Prefer the ggml-org URL.

2. What llama.cpp and llama-server are

The llama.cpp README describes the project's goal as enabling LLM (and multimodal LLM) inference with minimal setup on a wide range of hardware, locally and in the cloud. It is built on the ggml library.

The project README states that Apple Silicon is a first-class citizen and is optimized through ARM NEON, Accelerate, and Metal.

`llama-server` (also documented as `llama serve` after the official installer) is the project's HTTP server. Official server documentation says it provides an OpenAI-compatible API, a built-in chat app called Llama Chat, and multimodal support. You run it on your machine and talk to it from a browser or any client that can call those APIs.

3. Installing on a Mac

The official README lists several supported ways to get the software. Pick one. You do not need all of them.

Option A — Official installer

The README publishes:

```
curl -LsSf https://llama.app/install.sh | sh
```

Before running a pipe-to-shell command, open https://llama.app and review the instructions. After that installer is used, the README shows:

```
llama serve -hf ggml-org/Qwen3.5-0.8B-GGUF
```

The `-hf` form downloads a model from Hugging Face at run time. That is convenient, but it is a network call. See the privacy section if you want to avoid that.

Option B — Pre-built binaries

The README also says you can download pre-built binaries from the releases page. Releases include a macOS Apple Silicon (arm64) build. Download the archive that matches your Mac, unpack it, and run the `llama-server` binary from that folder. This avoids compiling, and you can keep the files entirely offline after the download.

Option C — Build from source

From `docs/build.md`:

```
git clone https://github.com/ggml-org/llama.cpp
cd llama.cpp
cmake -B build
cmake --build build --config Release
```

The server README says you can also build only the server target:

```
cmake -B build
cmake --build build --config Release -t llama-server
```

The binary is then at `./build/bin/llama-server`. The build guide says that on macOS, Metal is enabled by default, and that Metal runs computation on the GPU. To disable Metal at compile time, the guide documents `-DGGML_METAL=OFF`. When Metal is built in, you can still turn GPU inference off at run time with `--n-gpu-layers 0`.

You need the usual macOS developer tools (Xcode command-line tools, and CMake). This sample does not walk through installing those tools.

4. Choosing a GGUF file and thinking about RAM

A GGUF file is a packaged model (and, separately, a packaged multimodal projector). The llama.cpp README states that the project supports integer quantization at 1.5-bit, 2-bit, 3-bit, 4-bit, 5-bit, 6-bit, and 8-bit, “for faster inference and reduced memory use.”

Names such as `Q4_K_M` or `Q8_0` are quantization types. In general, a lower-bit quantization is a smaller file and uses less memory; a higher-bit quantization is closer to the unquantized weights and is a larger file. Quality is not only about the letter in the name, and this guide does not claim that one type is always best.

The official server help text says that when you use `-hf` and omit a quant, the default is `Q4_K_M`, or the first file in the repository if that type is not present. That is a documented default, not a recommendation for every machine.

RAM on Apple Silicon is unified. The model weights, the context (KV cache), the operating system, and any mmproj file share the same memory. A safe way to choose a file, without inventing a universal gigabyte chart, is:

1. Read the GGUF file size on the model’s download page. That size is a lower bound for the weights once loaded; it is not the whole story.
2. Leave headroom above that file size for the context window you will actually use. Larger `-c` / `--ctx-size` values use more memory.

3. If you load an mmproj file for vision, count that file as additional memory, plus working space for image tokens. Official multimodal notes say some models may need a large context window, for example `-c 8192`.
4. Leave headroom for macOS itself. If the process is killed or the machine starts swapping heavily, use a smaller quant, a shorter context, or a smaller model. This guide does not publish a table of "model X fits in Y GB," because that depends on the model, the quant, the context, the projector, and the software version.

Start with a small official example, such as the README's `ggml-org/Qwen3.5-0.8B-GGUF`, to confirm the server starts on your Mac. Then move to a larger local file if you want more capability and you have the memory for it.

5. Running llama-server from a local file

The official server README's Unix quick start is:

```
./llama-server -m models/7B/ggml-model.gguf -c 2048
```

Replace the path and the context size with your file and a context you can afford. The same document says the server listens by default on `127.0.0.1:9931`, and that you can open the web front end at that address. Documented defaults include:

Flag	Meaning in the official server README (as fetched)
`-m`	Path to the local GGUF model file (used in the quick-start example).
`-c, --ctx-size`	Prompt context size. Default 0 means "loaded from the model."
`--host`	Listen address. Default `127.0.0.1`.
`--port`	Listen port. Default `9931`.
`-ngl, --n-gpu-layers`	How many layers to store in GPU memory: a number, `auto`, or `all`. Default `auto`.
`--jinja` / `--no-jinja`	Whether to use the Jinja template engine for chat. Default: enabled.
`-hf`	Hugging Face repo, optional `:quant`. Downloads the model (and an mmproj if available) unless you disable that.

A local-file example that stays on your machine after the files are already downloaded:

```
./llama-server -m /path/to/model.Q4_K_M.gguf -c 4096 --host 127.0.0.1 --port 9931
```

Then open `http://127.0.0.1:9931` in a browser on the same Mac. If a model's card or the server log tells you that a chat template is required and replies look wrong, check the official flags for `--jinja` and `--chat-template` in the live server README. Jinja is enabled by default in the documentation fetched for this sample; you should not need to add `--jinja` unless you are on an older binary where the default differed.

6. Vision with an mmproj file

llama.cpp's multimodal documentation says image, audio, and video input are supported through `libmtmd`, and that `llama-server` exposes this via the OpenAI-compatible `/chat/completions` API.

There are two official ways to enable it:

1. Use `-hf` with a supported repo. The multimodal doc says the projector is handled for you; use `--no-mmproj` to load the same repo without vision, or `--mmproj local_file.gguf` to point at a local projector while still using `-hf`.
2. Use a local model and a local projector: `-m model.gguf` plus `--mmproj file.gguf`.

Official examples include:

```
llama-server -m gemma-3-4b-it-Q4_K_M.gguf --mmproj mmproj-gemma-3-4b-it-
Q4_K_M.gguf

llama-server -hf ggml-org/gemma-3-4b-it-GGUF
```

By default, the multimodal projector is offloaded to the GPU. The docs say you can disable that with `--no-mmproj-offload`.

The official list of pre-quantized multimodal GGUFs is on Hugging Face at the ggml-org collection linked from `docs/multimodal.md`:

https://huggingface.co/collections/ggml-org/multimodal-ggufs-68244e01ff1f39e5bebeeedc

That list includes vision models under families such as Gemma 3, SmolVLM, Pixtral, Qwen2-VL, Qwen2.5-VL, InternVL, and others. This sample does not claim that every third-party "abliterated" or fine-tuned vision GGUF will work. If you use a model that is not on the official list, check that model's card and the llama.cpp version it asks for. Some architectures only work after a specific llama.cpp release. If the server starts but rejects images, the usual causes are a missing or mismatched mmproj, or a build that is older than the model's required support.

Keep the mmproj file paired with the text model it was published for. Mixing a projector from one repo with weights from another is a common way to get a startup error or silent poor results.

7. Privacy tips

- **Stay on localhost.** The documented default listen address is `127.0.0.1`, which is only reachable from the same computer. Do not pass `--host 0.0.0.0` unless you intend to expose the server on a network and you understand that anyone who can reach that port may use the model.
- **Prefer local files when you want offline use.** `-m` and `--mmproj` with paths you already downloaded do not need Hugging Face at run time. `-hf` does.
- **Review what you paste.** A local server does not send your prompts to a vendor by default, but anything you paste into the chat is still on that machine. Do not paste passwords, API keys, wallet seeds, or other people's private data.
- **Treat model files as untrusted programs.** Download GGUF files from sources you have a reason to trust, and verify the repo and filename. This guide does not ask you to enable `trust_remote_code` or any equivalent remote-code path; llama-server loads GGUF weights, not arbitrary Python from a model card.
- **You are responsible for use.** Local models may generate disallowed, incorrect, or sensitive content. This is a technical setup guide, not legal advice.

8. If something fails

- **The binary is missing or won't run.** Confirm you downloaded an Apple Silicon (arm64) build, or that `./build/bin/llama-server` exists after a successful CMake build.
- **The model path is rejected.** Use an absolute path in `-m` and check that the file ends in `.gguf`.
- **The process is killed or the machine thrashes.** Use a smaller GGUF, reduce `-c`, and if you loaded vision, try without the mmproj first to see whether the projector is the extra load.
- **Port 9931 is already in use.** Pick another port with `--port`, or stop the other process. This guide does not tell you to stop software you did not start.
- **Images are not accepted.** Confirm you passed `--mmproj` (or used `-hf` with auto mmproj), that the projector matches the model, and that your llama.cpp build is new enough for that architecture. See `docs/multimodal.md`.
- **Replies look unformatted or ignore chat roles.** See the official notes on Jinja and `--chat-template` in the server README.

When you ask for help in public, say the llama.cpp release tag or commit, the exact command (with local paths shortened), the model filename, and the error text. You do not need to share your prompts.

9. Sources

1. ggml-org/llama.cpp README: https://github.com/ggml-org/llama.cpp and https://llama.app
2. Build guide, including the Metal section: https://github.com/ggml-org/llama.cpp/blob/master/docs/build.md
3. LLaMA.cpp HTTP Server: https://github.com/ggml-org/llama.cpp/blob/master/tools/server/README.md
4. Multimodal documentation: https://github.com/ggml-org/llama.cpp/blob/master/docs/multimodal.md
5. Official releases (macOS Apple Silicon): https://github.com/ggml-org/llama.cpp/releases
6. Official multimodal GGUF collection: https://huggingface.co/collections/ggml-org/multimodal-ggufs-68244e01ff1f39e5bebeeedc