

AI BENCH LAB

# Setting Up Local Providers

Configure Ollama, LM Studio, or the built-in runtime to benchmark models on your own hardware.

---

**Version 1.1** | April 2026

[aibenchlab.com](https://aibenchlab.com)

# Contents

## 1. Three Ways to Run Models Locally

---

## 2. Built-In llama.cpp Server

---

## 3. Ollama

---

## 4. LM Studio

---

## 5. Adding Providers Manually

---

## 6. The Model Catalog

---

## 7. Troubleshooting

---

*Configure Ollama, LM Studio, or the built-in runtime to benchmark models on your own hardware.*

## 1. Three Ways to Run Models Locally

AiBenchLab supports three local inference providers. You can use any combination simultaneously.

| Provider           | Protocol           | Default Port     | GPU Support   | Setup Required               |
|--------------------|--------------------|------------------|---------------|------------------------------|
| Built-in llama.cpp | OpenAI-compatible  | 8900–9999 (auto) | CUDA, Vulkan  | None — ships with the app    |
| Ollama             | Ollama (dedicated) | 11434            | Via Ollama    | Install Ollama separately    |
| LM Studio          | OpenAI-compatible  | 1234             | Via LM Studio | Install LM Studio separately |

## 2. Built-In llama.cpp Server

AiBenchLab includes its own llama.cpp inference server. It is fully managed by the app — no configuration, no terminal commands, no external software.

### How It Works

1. You select a local GGUF model (from the catalog or a file on disk).
2. The app launches a llama.cpp server instance for that model on an auto-assigned port (8900–9999).
3. The server exposes an OpenAI-compatible API at `http://127.0.0.1:<port>`.
4. When the benchmark completes (or you unload the model), the server shuts down.

Each model gets its own server instance. The app manages the full lifecycle — start, health check, error recovery, and shutdown.

### GPU Acceleration

During first-launch setup, the app downloads one of two backends:

| Backend | When Selected                   | Download Size | GPU Compatibility              |
|---------|---------------------------------|---------------|--------------------------------|
| CUDA    | NVIDIA GPU detected             | ~141 MB       | NVIDIA GPUs (CUDA 13.1+)       |
| Vulkan  | No NVIDIA GPU, or AMD/Intel GPU | ~55 MB        | AMD, Intel, NVIDIA (universal) |

If you have an NVIDIA GPU, the CUDA backend is selected automatically. If no compatible GPU backend is found at runtime, the server falls back to CPU inference and logs a warning.

## Server Configuration

When launching a model, the app configures:

| Parameter    | Default           | Notes                                       |
|--------------|-------------------|---------------------------------------------|
| Context size | 4,096 tokens      | Adjustable per model                        |
| GPU layers   | Auto (all layers) | Set to 0 for CPU-only                       |
| Host         | 127.0.0.1         | Localhost only — not exposed to the network |

## Error Recovery

The built-in server detects common failure modes and provides actionable guidance:

- **Chat template parsing error:** Retries with `--no-jinja --chat-template chatml`
- **CUDA / cuBLAS error:** Suggests reducing GPU layers or switching to Vulkan
- **Out of memory / insufficient VRAM:** Suggests a smaller model or quantization
- **Invalid GGUF magic:** Model file is corrupted — re-download recommended
- **Context size too large:** Reduce context size in model settings

## System Check

Go to **Settings > Components** to run a system check. The check verifies:

- llama.cpp binary is installed and executable
- GPU backend DLLs are present (CUDA or Vulkan)
- CUDA runtime libraries (Windows: `cudart64_*.dll`, `cublas64_*.dll`)
- Loaded backends (CPU, CUDA, Vulkan) via `--version` output
- GPU device detection

The result is a pass/fail summary with details on available backends and any missing components.

## The Judge Server

Separately from the model server above, AiBenchLab runs a **Judge Server** on port 8899 to score model responses. The judge uses **Gemma 3 4B Instruct Q8\_0** on a bundled, version-locked llama.cpp instance to ensure scoring consistency across releases. The judge runs on its own dedicated process so it doesn't compete with the model under test for GPU memory.

---

## 3. Ollama

Ollama is a standalone local inference server. AiBenchLab detects it automatically on startup.

### Setup

1. Install Ollama from [ollama.ai](https://ollama.ai).
2. Pull a model:

```
bash ollama pull llama3.2
```

1. Start the server (if not already running):

```
bash ollama serve
```

1. Launch AiBenchLab. Ollama is auto-detected at `http://localhost:11434`. Its models appear in the model selection panel of the Benchmark Wizard.

### Connection Details

| Setting        | Value                                                                                      |
|----------------|--------------------------------------------------------------------------------------------|
| Endpoint       | <code>http://localhost:11434</code>                                                        |
| API            | Ollama native ( <code>/api/tags</code> , <code>/api/show</code> , <code>/api/chat</code> ) |
| Authentication | None required                                                                              |
| Auto-detected  | Yes — checked on every app launch                                                          |

### Model Discovery

AiBenchLab fetches your installed models via `GET /api/tags`, then queries each model's metadata via `POST /api/show` to extract:

- Context window size
- Parameter count

- Vision capability (detected from model families: `clip`, `gemma3`, `mlama`, or names containing `vl / vision`)

## Multimodal Support

Vision-capable Ollama models (e.g., LLaVA, Gemma 3) are automatically detected and can run multimodal benchmark tests.

---

## 4. LM Studio

LM Studio provides a GUI for downloading and running local models with an OpenAI-compatible API.

### Setup

1. Install LM Studio from [lmstudio.ai](https://lmstudio.ai).
2. Download a model through the LM Studio interface.
3. Load the model and start the local server (LM Studio > Local Server > Start).
4. Launch AiBenchLab. LM Studio is detected at `http://localhost:1234/v1`.

### Connection Details

| Setting        | Value                                                                       |
|----------------|-----------------------------------------------------------------------------|
| Endpoint       | <code>http://localhost:1234/v1</code>                                       |
| API            | OpenAI-compatible ( <code>/models</code> , <code>/chat/completions</code> ) |
| Authentication | None required for localhost                                                 |
| Auto-detected  | Yes — health-check probed on startup                                        |

LM Studio may take a moment to become available after starting — the app confirms the connection is stable before listing it as ready. This accounts for LM Studio's model loading time.

### Model Discovery

Models are fetched via the standard OpenAI `GET /models` endpoint. Embedding models are automatically filtered out.

---

## 5. Adding Providers Manually

If a provider isn't auto-detected (e.g., running on a non-default port), add it manually:

1. Go to **Settings** in the sidebar.
2. Click **+ Add Provider**.
3. Select the provider template (Ollama, LM Studio, or Custom OpenAI-Compatible).
4. Edit the endpoint URL if needed.
5. Click **Save**, then **Test Connection**.

The test verifies connectivity, counts available models, and measures response time.

---

## 6. The Model Catalog

AiBenchLab includes a searchable catalog of over **51,000 HuggingFace models**, sorted by popularity. The catalog ships with a bundled seed database of the top 5,000 models for instant availability.

### Browsing Models

1. Click **Model Catalog** in the sidebar.
2. Use size filter buttons (Tiny, Small, Medium, Large, XL) for quick filtering.
3. Check **"Only show models that fit my hardware"** to filter by your GPU's VRAM.
4. Apply presets: - **Production Ready**: Commercial license, chat template, evaluation results - **Commercial Safe**: Legally safe for business use - **Air-Gapped**: Offline-capable local models - **Quick Explore**: Minimal filters

### GPU Fit Detection

The **GPU Fit** column shows whether a model fits your hardware:

| Status                      | Meaning                                |
|-----------------------------|----------------------------------------|
| Fits your GPU (green)       | Model fits within available VRAM       |
| Too large (red)             | Model exceeds GPU capacity             |
| May need offloading (amber) | Might work with partial CPU offloading |
| Unknown (gray)              | VRAM estimate not available            |

GPU detection uses `nvidia-smi` on NVIDIA systems and DXGI on Windows for other GPUs.

## Downloading Models

Click **Get** on any model card. The download routes through the appropriate handler:

- **GGUF models (HuggingFace):** Downloaded directly to your local models directory
- **Ollama models:** Triggers `ollama pull` if Ollama is running
- **LM Studio models:** Opens the download in LM Studio

## Refreshing the Catalog

Click **Refresh Sources** and choose a sync limit:

| Limit         | Duration | Description                |
|---------------|----------|----------------------------|
| 5,000         | ~1 min   | Fast sync                  |
| 10,000        | ~2 min   | Medium sync                |
| <b>20,000</b> | ~5 min   | <b>Recommended default</b> |
| 50,000        | ~10 min  | Full catalog               |

Models are synced by download count — you always get the most popular models first.

## 7. Troubleshooting

### Ollama not detected

- Verify Ollama is running: `ollama list` should return your models.
- Check the port: `curl http://localhost:11434/api/tags` should return JSON.
- If using a non-default port, add the provider manually in Settings.

### LM Studio not detected

- Ensure the local server is running (LM Studio > Local Server > Start).
- Verify a model is loaded — LM Studio won't respond to API calls without a model.
- Check the port: `curl http://localhost:1234/v1/models` should return JSON.

### Built-in server fails to start

- Run the system check in **Settings > Components**.
- If CUDA errors appear, try switching to the Vulkan backend in Components.

- For "out of memory" errors, try a smaller model or reduce GPU layers.
- Check that no other application is using ports in the 8900–9999 range.

## Models not appearing

- Verify the provider shows a green connection status in Settings.
  - Click **Refresh** on the provider card to re-scan for models.
  - For Ollama, ensure you've pulled at least one model: `ollama pull llama3.2`.
- 

## Resources

- Documentation: [aibenchlab.com/docs](https://aibenchlab.com/docs)
- Support: [aibenchlab.com/contact](https://aibenchlab.com/contact)
- YouTube walkthroughs: [youtube.com/@aibenchlab](https://youtube.com/@aibenchlab)
- Email: [support@aibenchlab.com](mailto:support@aibenchlab.com)

*AiBenchLab is built by a solo founder with 40+ years of software engineering experience. Every feature is designed with the conviction that you deserve honest, transparent, and reproducible AI model evaluations. No hype. Just truth.*