TRAINEE HANDBOOK

Working with Local LLMs, Offline

A practical, step-by-step guide to running open-source language models on your own computer with LM Studio and Ollama — and using them for real daily-work tasks with Retrieval-Augmented Generation (RAG).

LM Studio | Ollama | RAG | AnythingLLM | Fine-Tuning Basics

Prepared for internal training use

August 2026

Table of Contents

Twelve sections, designed to take you from zero to comfortably productive with offline AI.

SECTION 01

Why Local LLMs?

A **local LLM** is a large language model that runs entirely on your own laptop or desktop — no internet connection required once it is downloaded, and no data ever leaves your machine. This is different from tools like ChatGPT or Claude's web app, where every message you type travels to a company's servers to be processed.

For day-to-day work, running a model locally has four practical advantages:

Reason	What it means for you
Privacy	Confidential reports, internal policies, HR data, or client information never leave your device. Nothing is logged on an external server.
Offline access	Once the model and your documents are loaded, you can work on a train, in a basement server room, or anywhere without Wi-Fi.
Zero recurring cost	No per-message or per-token billing. You pay once in electricity and hardware, not per query.
Full control	You choose the exact model, you can inspect how it was configured, and you can adapt it to your own documents and writing style.

> **NOTE**
>
> Local models are typically smaller than the largest cloud models, so for extremely complex reasoning tasks a cloud assistant may still outperform them. Local LLMs shine at everyday, repeatable tasks — drafting, summarizing, searching your own documents, and answering questions about material you feed them.

By the end of this handbook you will be able to install and run a local model, connect it to your own files so it can answer questions grounded in them, and know when (and how) to go one step further and actually train a model on your own data.

SECTION 02

Core Concepts: RAG vs. Fine-Tuning

There are two fundamentally different ways to make an LLM "know" something it wasn't originally trained on:

Retrieval-Augmented Generation (RAG)

The model itself is **never changed**. Instead, at the moment you ask a question, a search system finds the most relevant chunks of your documents and hands them to the model as context, along with your question. The model then answers using that context — much like an open-book exam.

Fine-Tuning

The model's internal weights are **actually adjusted** through additional training, so the behaviour itself changes — its tone, structure, or domain vocabulary becomes different going forward, even without any extra context supplied at question time.

Which one do you need?

Choose RAG if…	Choose Fine-Tuning if…
You want answers drawn from your PDFs, notes, policies, or reports	You want a specific, consistent output *style* (e.g. "always answer like a senior HR officer")
Your source material changes often	You have a stable, curated set of question/answer examples
You need traceability — "which document did this answer come from?"	You're comfortable re-training periodically as knowledge changes
You want to get started today, with no coding	You have access to a GPU with sufficient memory

RULE OF THUMB

Start with RAG. It is faster to set up, requires no GPU training, and covers the vast majority of "help me use my own documents" use cases. Only move to fine-tuning once you have a clear, repeatable style or format you want baked permanently into the model — and ideally, combine both: fine-tune for style, RAG for up-to-date facts.

This handbook therefore teaches RAG first (Sections 4–7), since that is the right starting point for almost everyone, and covers fine-tuning as an optional advanced path in Section 9.

SECTION 03

Choosing a Model for Your Hardware

Open-source models come in different **sizes** (measured in billions of parameters, e.g. "7B", "8B") and different **file formats**. For local use, the two formats you'll encounter are:

- **GGUF** — a compressed, quantized format used by LM Studio and Ollama. This is what you'll use almost all the time.
- **Safetensors / full-precision weights** — used mainly during training or fine-tuning, before conversion to GGUF.

Recommended starting models

Model	Size	Good for	Minimum RAM (quantized)
Llama 3.1 8B Instruct	8B	General assistant, writing, Q&A	~8 GB
Mistral 7B Instruct	7B	Fast, efficient general use	~8 GB
Qwen 2.5 7B Instruct	7B	Strong multilingual & reasoning	~8 GB
Phi-3.5 Mini	3.8B	Low-spec laptops, quick answers	~4 GB

HOW TO READ "QUANTIZATION"

You'll see model names like `Q4_K_M` or `Q8_0`. This is how much the model has been compressed. `Q4` variants are smaller and faster but slightly less precise; `Q8` or `Q6` variants are larger, slower, and closer to full quality. For a laptop with 16 GB RAM, a 7–8B model at `Q4_K_M` is the sweet spot.

Your machine has…	Recommended model size
8 GB RAM, no dedicated GPU	3–4B models (e.g. Phi-3.5 Mini), Q4 quantization
16 GB RAM, no/entry GPU	7–8B models, Q4_K_M quantization
16–24 GB VRAM GPU	7–13B models at higher quality (Q6–Q8), faster responses
24 GB+ VRAM GPU	13–34B models, or multiple 7–8B models loaded simultaneously

Both LM Studio and Ollama show you compatibility warnings before you download a model that's too large for your machine, so you don't need to calculate this precisely — start with an 8B model and adjust from there.

SECTION 04

Setting Up LM Studio

LM Studio is a free desktop application with a graphical interface — the easiest way to download, chat with, and serve local models, especially if you prefer not to use the command line.

4.1 Installation

1. Go to **lmstudio.ai** and download the installer for your operating system (Windows, macOS, or Linux).
2. Run the installer and open LM Studio.
3. On first launch, LM Studio will offer to download a starter model — you can accept this, or skip it and choose your own in the next step.

4.2 Downloading and loading a model

1. Click the **search / discover** icon (magnifying glass) in the left sidebar.
2. Search for a model, e.g. `Llama 3.1 8B Instruct` or `Mistral 7B Instruct`.
3. Choose a **GGUF** version with a quantization such as `Q4_K_M` (a good balance of speed and quality for most laptops).
4. Click **Download** and wait for it to complete (model files are typically 4–8 GB).
5. Go to the **chat** icon, select your downloaded model from the dropdown at the top, and click **Load**.

4.3 Having your first conversation

Once a model is loaded, simply type into the chat box at the bottom of the screen. You can adjust settings such as **temperature** (creativity — lower is more focused, higher is more varied) and **system prompt** (instructions that apply to every message, e.g. "You are a helpful assistant for HR queries. Answer concisely.") from the right-hand panel.

4.4 Turning on the Local Server (needed for RAG)

To let other applications (like the RAG tool in Section 6) talk to your model, you need to expose it as a local API server:

1. Click the **Developer** tab (icon that looks like `</>`) in the left sidebar.
2. Toggle **Start Server** to ON.
3. Confirm the server address shown, typically `http://localhost:1234/v1`.
4. Leave LM Studio running in the background — other tools will now be able to send requests to this address.

> **TIP**

> This local server uses the same request format as OpenAI's API, so almost any RAG tool, coding plugin, or script that supports "custom OpenAI-compatible endpoint" will work with LM Studio out of the box.

SECTION 05

Setting Up Ollama

Ollama is a lightweight, command-line-first tool for running local models. It's the preferred option if you're comfortable typing a few commands, want the fastest possible setup, or plan to connect several different apps to the same running model.

5.1 Installation

1. Go to **ollama.com** and download the installer for your operating system.
2. Run the installer. Ollama installs a background service that starts automatically.
3. Open a terminal (Command Prompt, Terminal, or PowerShell) to confirm it's working: `ollama --version`

5.2 Downloading and running a model

```
# Download (pull) a model
ollama pull llama3.1:8b

# Start an interactive chat session in the terminal
ollama run llama3.1:8b
```

The first time you run a model it downloads automatically if not already pulled. Once loaded, you can type directly into the terminal and get responses. Type `/bye` to exit the chat.

5.3 Useful everyday commands

Command	What it does
`ollama list`	Shows every model you've downloaded
`ollama rm llama3.1:8b`	Deletes a model to free disk space
`ollama ps`	Shows which models are currently loaded in memory
`ollama stop llama3.1:8b`	Unloads a model from memory

5.4 Customizing behaviour with a Modelfile

You can give a model a permanent system prompt and default settings by creating a small text file called a **Modelfile**:

```
FROM llama3.1:8b
PARAMETER temperature 0.3
SYSTEM """
You are a helpful internal assistant. Answer clearly and concisely,
using bullet points for multi-part answers.
"""
```

Save this as `Modelfile` (no extension), then create your custom version:

```
ollama create my-assistant -f Modelfile
ollama run my-assistant
```

> **NOTE**
>
> Ollama exposes its own API at `http://localhost:11434`, which — like LM Studio's server — most RAG tools can connect to directly by selecting "Ollama" as the provider.

SECTION 06

Building a RAG Assistant with AnythingLLM

This is where local LLMs become genuinely useful for daily work: connecting a model to your own documents so it can answer questions grounded in them, with references back to the source. We'll use **AnythingLLM**, a free desktop app built specifically for this.

6.1 Install AnythingLLM

1. Download AnythingLLM Desktop from its official site and install it like any other application.
2. Open the app. You'll be asked to choose an **LLM Provider** — this is the model that will generate answers.

6.2 Connect it to LM Studio or Ollama

If using LM Studio

- LLM Provider: **LM Studio**
- Base URL: `http://localhost:1234/v1`
- Make sure the LM Studio server is running (Section 4.4)

If using Ollama

- LLM Provider: **Ollama**
- Base URL: `http://localhost:11434`
- Make sure Ollama is running in the background

6.3 Choose an embedding model and vector database

Two extra pieces make RAG work behind the scenes — you generally only need to accept the defaults on first use:

- **Embedding model** — converts text into numbers that capture meaning, so similar ideas can be matched. AnythingLLM's built-in default is fine to start.
- **Vector database** — stores those numbers for fast searching. Choose the built-in **LanceDB** option; it requires no separate installation.

6.4 Create a Workspace and upload documents

1. Click **New Workspace** and give it a name (e.g. "HR Policies" or "Training Materials").
2. Click the upload icon and add your files — PDFs, Word documents, plain text, CSV, or even pasted webpages.
3. Click **Save and Embed**. AnythingLLM will automatically split your documents into chunks, generate embeddings for each chunk, and store them in the vector database.
4. Wait for the embedding progress bar to finish — this can take a few seconds to a few minutes depending on document size.

6.5 Ask questions

Open the chat for your workspace and ask a question in plain language, for example: *"What is the process for requesting annual leave?"* Behind the scenes, AnythingLLM:

1. Searches the vector database for the most relevant chunks of your uploaded documents.

2. Builds a combined prompt containing those chunks plus your question.
3. Sends it to your local model (via LM Studio or Ollama).
4. Returns an answer, usually with a citation back to the source document.

> **COMMON PITFALL**
>
> If answers seem generic or ignore your documents, check that you actually clicked **Save and Embed** after uploading, and that you're chatting inside the correct workspace — each workspace has its own separate set of documents.

SECTION 07

Daily Use Cases & Example Workflows

Once your model (and optionally a RAG workspace) is set up, here are practical, ready-to-use applications you can start with today.

Document Q&A Ask questions about long reports or policies

Upload a lengthy policy manual, contract, or research report into a RAG workspace, then ask targeted questions instead of reading the whole document.

EXAMPLE PROMPT

```
Summarize the key eligibility rules for promotion described in this document,
and list any exceptions mentioned.
```

Writing Drafting emails, memos, and reports

Use the plain chat (no RAG needed) to draft first versions of routine communication, which you then review and personalize.

EXAMPLE PROMPT

```
Draft a short, polite email to a vendor about a 15-day payment delay,
requesting an updated timeline. Keep it under 120 words.
```

Summarization Condensing meeting notes or long threads

Paste raw meeting notes or a long email thread and ask for a structured summary.

EXAMPLE PROMPT

```
Summarize these meeting notes into: (1) decisions made, (2) action items with
owners, and (3) open questions. [paste notes]
```

Document Q&A Internal knowledge base / FAQ assistant

Upload FAQs, onboarding guides, or SOPs into a shared RAG workspace so new team members can ask questions instead of searching through folders.

Analysis First-pass analysis of data or reports

Paste a data table or short report excerpt and ask for patterns, risks, or a structured breakdown — useful as a starting point before deeper manual review.

EXAMPLE PROMPT

```
Review this quarterly summary and outline: Assumptions -> Risks ->
Notable Changes -> Open Questions.
```

Learning Training material & explanation generation

Ask the model to explain a concept at a chosen level of detail, or turn a technical document into simpler training notes for a new employee.

Coding Code help, without sending code to the cloud

Both LM Studio and Ollama work well with coding-oriented models (e.g. Qwen2.5-Coder) for explaining, debugging, or writing small scripts entirely offline — useful when working with sensitive or proprietary codebases.

TIP

Keep a running "prompt library" document of the instructions that work well for your team's recurring tasks. Reusing a well-worded prompt is often more effective than writing from scratch every time.

SECTION 08

Hands-On Practice Exercises

Work through these in order. Each builds on the setup from earlier sections.

Exercise 1 — First Chat

Install LM Studio or Ollama (your choice), load an 8B model, and ask it to explain your job role in three sentences, as if to a new colleague. Notice the response speed and quality.

Exercise 2 — Prompting Practice

Ask the same question three different ways: (a) as a one-line question, (b) with a role instruction ("You are a senior analyst…"), (c) with a required output format ("Answer in exactly 3 bullet points"). Compare the three answers.

Exercise 3 — Build Your First RAG Workspace

Install AnythingLLM, connect it to your running model, and upload 2–3 documents you use regularly (a policy, a report, or your own notes). Ask five real questions you would normally have to search the document manually to answer. Check whether the citations point to the right source.

Exercise 4 — Compare RAG vs. No-RAG

Ask the same question both inside your RAG workspace and in a plain chat (no documents attached). Note the difference in accuracy and specificity.

Exercise 5 — Draft-and-Refine

Use the model to draft a short document you actually need this week (an email, a summary, a short report section). Edit the draft, and record what you changed — this is useful input if your team later decides to fine-tune a model on your preferred style.

SECTION 09

Going Further: Fine-Tuning Your Own Model

Fine-tuning changes a model's actual weights so a particular style or domain vocabulary becomes built in, without needing to supply context every time. It is optional, more technical, and best attempted after your team is already comfortable with RAG.

9.1 When it's worth it

- You need consistent structure or tone across many outputs (e.g. always "Assumptions → Risks → Recommendation").
- You have 200–10,000 high-quality example question/answer or instruction/response pairs.
- You have access to a GPU with at least 8–24 GB of VRAM (for 7–8B models using an efficient method called QLoRA).

> **IF YOU ONLY HAVE DOCUMENTS, NOT CURATED EXAMPLES**
>
> Stick with RAG. Fine-tuning needs structured training *examples*, not raw source documents — raw documents are exactly what RAG is designed to use as-is.

9.2 Preparing a training dataset

Data is stored as JSON Lines (`.jsonl`), one example per line:

```
{"instruction": "Explain the key provisions of Rule X.",
 "input": "",
 "output": "Rule X states that ..."}
{"instruction": "Draft a short email to a vendor about a delayed payment.",
 "input": "Vendor: ABC Ltd, Delay: 15 days",
 "output": "Subject: Payment Update Request\n\nDear ABC Ltd, ..."}
```

Guidelines: remove timestamps, headers, and irrelevant metadata; keep each example focused (roughly 200–600 tokens); include a mix of 10–20% unusual-but-important edge cases; and split the set into training (~85%) and validation (~15%) files.

9.3 Training with Unsloth (QLoRA)

Unsloth is currently one of the most accessible tools for fine-tuning locally on a single GPU:

```
python -m venv finetune-env
source finetune-env/bin/activate   # Windows: finetune-env\Scripts\activate

pip install "unsloth[cu121-torch240] @ git+https://github.com/unslothai/unsloth.git"
pip install trl peft accelerate bitsandbytes datasets
```

A training script loads a base model (e.g. Llama 3.1 8B Instruct), your dataset, and a QLoRA configuration (rank 8–16, alpha 16–32, 4-bit quantization, 1–3 training epochs, learning rate around `2e-4`), then trains and monitors both training and validation loss — stopping early if validation loss starts rising.

9.4 Exporting and using your fine-tuned model

1. Merge the trained LoRA adapters into the base model.
2. Convert the merged model to GGUF format so LM Studio and Ollama can run it.
3. In Ollama, create a Modelfile pointing at your new `.gguf` file and run `ollama create my-model -f Modelfile`.
4. In LM Studio, load the `.gguf` file directly from Models > Load Model.

> **BEST OF BOTH WORLDS**
>
> Many teams fine-tune once for consistent tone and structure, then still layer RAG on top so the model's factual answers stay current as documents change — you don't have to choose only one.

SECTION 10

Troubleshooting & FAQ

Problem	Likely cause & fix
Model responses are very slow	Model is too large for your RAM/VRAM. Try a smaller model (e.g. 7B instead of 13B) or a more compressed quantization (Q4 instead of Q8).
AnythingLLM can't connect to LM Studio/Ollama	Confirm the server toggle is ON in LM Studio (Section 4.4), or that Ollama is running in the background. Check the base URL matches exactly.
RAG answers ignore my documents	Make sure documents were embedded (Section 6.4) and that you're chatting inside the correct workspace.
Answers are generic or vague	Add more specific instructions to your prompt, including desired format, length, and role.
Model gives an "out of memory" error	Close other memory-heavy applications, or switch to a smaller model / more aggressive quantization.
Downloaded model doesn't appear	Restart LM Studio/Ollama, or check the model finished downloading fully before trying to load it.
Which is better, LM Studio or Ollama?	LM Studio suits those who prefer a visual interface; Ollama suits those comfortable with the command line and wanting a lightweight background service. Both work equally well with AnythingLLM.

SECTION 11

Glossary of Terms

LLM (Large Language Model)

An AI model trained on large amounts of text, able to understand and generate human-like language.

Local LLM

An LLM that runs entirely on your own device rather than a remote server.

RAG (Retrieval-Augmented Generation)

A technique where relevant document chunks are retrieved and given to the model as context at question time, without changing the model itself.

Fine-tuning

Additional training that adjusts a model's internal weights, changing its behaviour permanently.

GGUF

A compressed file format for running LLMs efficiently on regular computers, used by LM Studio and Ollama.

Quantization

A compression technique that reduces model size and memory needs, at a small cost to precision (e.g. Q4, Q6, Q8).

Embedding

A numerical representation of text that captures its meaning, used to find similar content.

Vector database

A database optimized for storing and searching embeddings (e.g. LanceDB, Chroma).

System prompt

An instruction given to the model that applies to the entire conversation, setting its role or behaviour.

Temperature

A setting controlling response randomness — lower values give more focused, predictable answers.

QLoRA

An efficient fine-tuning method that trains a small set of additional parameters on top of a quantized base model, requiring far less memory than full fine-tuning.

Modelfile

A small configuration file used by Ollama to customize a model's default system prompt and parameters.

Token

A chunk of text (often close to a word or part of a word) that a model processes; usage and speed are often measured in tokens.

SECTION 12

Quick-Reference Cheat Sheet

LM Studio

1. Discover tab -> search model -> Download
2. Chat tab -> select model -> Load
3. Developer tab -> Start Server
   -> http://localhost:1234/v1

Ollama

ollama pull llama3.1:8b
ollama run llama3.1:8b
ollama list
ollama ps
ollama create my-model -f Modelfile

AnythingLLM (RAG)

1. New Workspace
2. Settings -> LLM Provider -> LM Studio / Ollama
3. Upload documents
4. Save and Embed
5. Ask questions in workspace chat

Good prompt structure

Role: "You are a ..."
Task: what to do
Format: bullets / word limit
Context: paste text or reference doc

WHERE TO GO NEXT

Revisit Section 8's exercises with your own real documents and tasks. Once your team is comfortable with day-to-day RAG use, Section 9 is there when you're ready to explore fine-tuning for a consistent house style.