Installing dependencies — Linux · macOS · Windows

> **This project runs with the language toolchains directly — there is no Docker.** You install a few tools once and run lessons with `./run -l <N>`. This guide covers **Linux, macOS, and Windows**, and all three supported developer stacks: **Python, Node.js, and C#**.

PDFs of this guide and every lesson live in `docs/pdf/` and are linked from the course site.

---

0. How it runs (no Docker) — and the polyglot model

The lessons are plain programs you run with your language's toolchain. Docker is **not** used.

This course is **polyglot** (Option B). The **Python** implementation is the reference and runs every implemented lesson today. Foundational lessons are also being implemented in **Node.js** and **C#**, selectable with a `--lang` flag:

```
./run -l 1                # Python (default / reference)
./run -l 1 --lang node    # Node.js implementation   (where available)
./run -l 1 --lang csharp  # C# / .NET implementation (where available)
```

If a language isn't ported for a lesson yet, `./run` tells you and points to the Python reference. See each lesson's **“Dependencies & Installation”** section for its language availability.

---

1. Base prerequisites (every lesson)

You always need **Git**, plus the toolchain(s) for the language(s) you want to use.

Git

- **Linux (Debian/Ubuntu):** `sudo apt update && sudo apt install -y git`
- **macOS:** `brew install git` · **Windows:** `winget install -e --id Git.Git`

Get the code

```
git clone https://github.com/nikolareljin/local-ai-lab.git
cd local-ai-lab
```

Python 3.10+ (reference stack — recommended for everyone)

- **Linux (Debian/Ubuntu):** `sudo apt install -y python3 python3-venv python3-pip`
- **Linux (Fedora):** `sudo dnf install -y python3 python3-pip`
- **macOS:** `brew install python`
- **Windows:** `winget install -e --id Python.Python.3.12` (or python.org — tick **Add to PATH**)

Node.js 18+ (only for `--lang node`)

- **Linux:** NodeSource or `sudo apt install -y nodejs npm`
- **macOS:** `brew install node` · **Windows:** `winget install -e --id OpenJS.NodeJS.LTS`

.NET 8 SDK (only for `--lang csharp`, and Lesson 6)

- **Linux:** Microsoft per-distro guide
- **macOS:** `brew install dotnet@8` · **Windows:** `winget install -e --id Microsoft.DotNet.SDK.8`

Verify what you installed:

```
git --version
python3 --version    # Windows: python --version
node --version       # if using --lang node
dotnet --version     # if using --lang csharp
```

---

2. Set up the language you'll use

Python (reference)

Linux / macOS

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Windows — PowerShell

```
python -m venv venv
venv\Scripts\Activate.ps1      # if blocked: Set-ExecutionPolicy -Scope CurrentUser RemoteSigned
pip install -r requirements.txt
```

Windows — cmd

```
python -m venv venv
venv\Scripts\activate.bat
pip install -r requirements.txt
```

`requirements.txt` covers Lessons 1–2: `pypdf`, `python-docx`, `rank-bm25`, `numpy`, `flask`, `requests`, `python-dotenv`, `mcp`. (`./run -l 1` also does this automatically on first use.)

Node.js (for ported lessons)

From the lesson's Node directory (e.g. `node/`):

```
npm install          # installs that lesson's package.json dependencies
```

C# / .NET (for ported lessons, and Lesson 6)

From the lesson's .NET project (e.g. `dotnet/`):

```
dotnet restore       # restores NuGet packages
dotnet build
```

---

3. The default AI: Claude Code CLI (no API key)

Every stack defaults to the **Claude Code CLI** — it uses your existing Claude Code login, so there is **no API key to manage**.

All platforms (via npm — needs Node.js 18+):

```
npm install -g @anthropic-ai/claude-code
claude                # first run signs you in
claude --version
```

> See the Claude Code docs for native installers. If `claude` isn't installed, the app says so and you can pick another provider (below).

---

4. Optional AI providers

Only needed if you set `RAG_PROVIDER` to something other than `claude`. Copy `.env.example` to `.env` for keys.

Ollama (fully local; also embeddings) - **Linux:** download the official script, **inspect it, then run** (don't pipe a remote script straight into a shell): `bash curl -fsSL https://ollama.com/install.sh -o ollama-install.sh less ollama-install.sh # review before running sh ollama-install.sh` (or follow the official Linux install docs) - **macOS:** `brew install ollama` (or download) - **Windows:** installer

```
ollama pull llama3.1           # chat / function calling
ollama pull nomic-embed-text   # embeddings (RAG_RETRIEVER=embeddings)
```

Gemini — key from aistudio.google.com → `.env`: `GEMINI_API_KEY=...` **OpenAI** — key from platform.openai.com → `.env`: `OPENAI_API_KEY=...`

---

5. Per-lesson dependencies & language availability

Lesson	Languages	Extra dependencies	Install
1 · RAG	Python ✓ · Node (planned) · C# (planned)	base only	`pip install -r requirements.txt`

2 · MCP	Python ✓ · Node (planned) · C# (planned)	`mcp` (in requirements) + Claude Code	§2 + §3
3 · LangChain	Python	LangChain + a vector store	`pip install langchain langchain-community langchain-ollama langchain-openai faiss-cpu`
4 · LangGraph	Python	LangGraph	`pip install langgraph langchain`
5 · Ollama + Function Calling	Python · Node (planned)	Ollama + a tool-capable model	§4 (Ollama) + `ollama pull llama3.1`
6 · Semantic Kernel	**C#/.NET**	.NET 8 SDK + SK NuGet	§1 (.NET) + `dotnet add package Microsoft.SemanticKernel`
7 · AWS Bedrock Agents	Python	AWS CLI + boto3	AWS CLI + `pip install boto3` + `aws configure`
8 · Google ADK	Python	`google-adk` + Gemini key	`pip install google-adk` + §4 (Gemini)

Legend: ✓ available · (planned) planned (Option B port in progress) · blank = single-language by nature.

---

6. Verify your setup

```
./run -l 1 test        # Lesson 1 (RAG) tests (Python)
./run -l 2 test        # Lesson 2 (MCP) tests
./run -l 1             # launch the RAG web UI
```

Windows note: `./run` is a Bash script — use **Git Bash** or **WSL**, or call the app directly:

```
python -m localrag web                 # Lesson 1 web UI
python -m localrag ask "your question" # Lesson 1 one-shot
python mcp_server.py                   # Lesson 2 MCP server (stdio)
pytest -q                              # all tests
```

Course: nikolareljin.github.io/local-ai-lab · Source: github.com/nikolareljin/local-ai-lab