

warning "Backend update (October 2026)" GitHub Models inference was retired on July 30, 2026. Any references below are retained only for historical context while this lab is migrated. Use Lab 013 ? Free LLM backends to configure Foundry Local or Ollama before continuing.

Lab 015: Ollama ? Run LLMs Locally for Free

Level: L100 Path: All paths Time: ~30 min ? Cost: Free ? Runs on your machine, no cloud, no API key

tip "Also try Foundry Local" Microsoft Foundry Local is an alternative to Ollama with an OpenAI-compatible API. See Lab 078: Foundry Local for a hands-on guide.

What You'll Learn

Install and run Ollama to serve LLMs locally Run Phi-4 (Microsoft's powerful small model) and Llama 3.2 on your own machine Generate text embeddings locally with nomic-embed-text Call Ollama from Python and C# using the OpenAI-compatible API Use Ollama as the LLM backend for Semantic Kernel (no API key needed)

Introduction

Ollama is an open-source tool that makes running LLMs on your laptop as easy as `ollama run phi4`. No API key, no cloud account, no usage costs ? just your own hardware.

This is valuable for: Privacy: sensitive data never leaves your machine Offline development: works without internet Cost control: zero API costs during development

Learning: experiment freely without worrying about bills

info "Hardware requirements" Ollama works on Mac (Apple Silicon or Intel), Windows, and Linux. For best performance: 16GB+ RAM. Works with 8GB but slower. GPU is optional ? models run on CPU too (just slower).

Prerequisites Setup

Install Ollama

1. Go to ollama.com and download the installer for your OS 2. Install and verify:

[diagram or code sample]

Ollama runs as a background service on `http://localhost:11434`.

tip "Quick Start with GitHub Codespaces" Open in GitHub Codespaces

All dependencies are pre-installed in the devcontainer.

? Supporting Files

note "Download these files before starting the lab" Save all files to a `lab-015/` folder in your working directory.

File | Description | Download | Model file | Ollama model configuration | ? Download | `chat_starter.py` | Starter script with TODOs | ? Download |

Lab Exercise

Step 1: Run your first model

[diagram or code sample]

This downloads Phi-4 (~9GB) on first run, then starts an interactive chat.

[diagram or code sample]

Other models to try:

[diagram or code sample]

Check what you have installed: [diagram or code sample]

Step 2: Pull an embedding model

[diagram or code sample]

This gives you a free local embedding model ? perfect for RAG without any API costs.

Step 3: Call Ollama from Python

Ollama's API is 100% OpenAI-compatible, so the same code that calls GitHub Models or Azure

OpenAI works here:

[diagram or code sample]

Step 4: Generate embeddings locally

[diagram or code sample]

Step 5: Use Ollama with Semantic Kernel

Because Ollama is OpenAI-compatible, plugging it into Semantic Kernel is trivial:

```
=== "Python"
```

[diagram or code sample]

```
=== "C#"
```

[diagram or code sample]

Step 6: Use Ollama as an MCP server backend

Since Ollama is OpenAI-compatible, any MCP server that calls an LLM can use it locally.

Just swap the client configuration:

[diagram or code sample]

No other code changes needed.

Step 7: Ollama via REST API directly

You can also call Ollama's native API (not OpenAI-compatible):

[diagram or code sample]

? Starter Files

Two files are provided to help you follow along:

[diagram or code sample]

The ? Modelfile creates a custom OutdoorGear Advisor persona on top of Phi-4. The ? chat_starter.py has 5 exercises covering basic completion, custom models, comparison, and streaming.

Model Comparison (on a typical laptop)

Model	Size	RAM needed	Speed	Quality	phi4-mini	2.5GB	4GB	???	Fast	Good	
llama3.2:1b	1.3GB	4GB	???	Very fast	Basic	llama3.2	2.0GB	6GB	??	Fast	Good
phi4	9.1GB	12GB	?	Moderate	Excellent	mistral	4.1GB	8GB	??	Fast	Very good
deepseek-r1	4.7GB	8GB	?	Moderate	Best reasoning						

Summary

You now have a fully local LLM stack:

? Ollama serving models on localhost:11434 ? Phi-4 (or Llama) for chat/reasoning ? free, private, offline ? nomic-embed-text for embeddings ? free, local ? Same code works for Ollama, GitHub Models, and Azure OpenAI ? just change base URL

Next Steps

Build a RAG app with local embeddings: ? Lab 022 ? RAG with GitHub Models + pgvector Use with Semantic Kernel plugins: ? Lab 023 ? SK Plugins, Memory & Planners Production local AI: ? Lab 044 ? Phi-4 + Ollama in Production

Supporting kit

The site build generates stable PDF and ZIP downloads. The ZIP contains the lab guide and every supporting file declared for this lab.

Download generated PDF Download generated ZIP

Supporting file: chat_starter.py

"" Lab 015: Ollama ? Local LLMs ===== Starter file ? chat with Ollama running locally. 100% free, no API keys, no internet required after model download.

Prerequisites: 1. Install Ollama: <https://ollama.com> 2. Pull a model: ollama pull phi4 3.

pip install openai (Ollama uses OpenAI-compatible API)

```

def streaming_chat(prompt: str, model: str = "phi4"): """Stream the response token by token."""
    print(f"\n? Streaming from {model}:\n")
    stream = ollama_client.chat.completions.create( model=model, messages=[{"role": "user",

```

```
"content": prompt}], stream=True, )
for chunk in stream: delta = chunk.choices[0].delta.content if delta: print(delta, end="",
flush=True) print("\n") # newline after streaming
???????????????????????????????????????????????????????????????????????????????? Main
???????????????????????????????????????????????????????????????????????????????? if __name__
== "__main__": # Step 1: Check what's available available = list_local_models()
if not available: print("\n? Quick start:") print(" 1. Install Ollama: https://ollama.com")
print(" 2. Run: ollama pull phi4") print(" 3. Run this script again") else: # Step 2:
Simple question print("\n=== Exercise 2: Simple Completion ===") reply = ask_ollama(
"What's the most important piece of gear for a first-time hiker?", model=available[0] )
print(reply)
# Step 3: Streaming print("\n=== Exercise 5: Streaming ===") streaming_chat( "Explain the
layering system for outdoor clothing in 3 steps.", model=available[0] )
# Step 4: Custom OutdoorGear model (requires Modelfile setup) print("\n=== Exercise 3:
Custom OutdoorGear Model ===") print("Create the model first: ollama create outdoorgear -f
Modelfile") # Uncomment to run: chat_with_outdoorgear()
# Step 5: Compare models (if you have multiple) if len(available) >= 2: print("\n===
Exercise 4: Compare Models ===") compare_local_models( available[:2], "What sleeping bag
temperature rating do I need for 0°C camping?" )
```