Lesson 8 · A Stateful Agent with LangGraph

PDF: this lesson · **Install (Linux · macOS · Windows):** guide · PDF

> **Part of local-ai-lab** - a hands-on course for building local AI.

> **Course home:** https://nikolareljin.github.io/local-ai-lab/ **Source:** https://github.com/nikolareljin/local-ai-lab

> **Lessons:** 1 · RAG → 2 · MCP → 3 · Hybrid retrieval → 4 · RAG safety → 5 · RAG evaluation → 6 · Repo assistant → 7 · LangChain → **8 · LangGraph (you are here)** → 9 · Ollama tools → 10 · Semantic Kernel → 11 · Bedrock Agents → 12 · Google ADK → ... → 15 · Docs from changes

> **Status: planned.** Outline below; full step-by-step coming later. Star the repo to follow along.

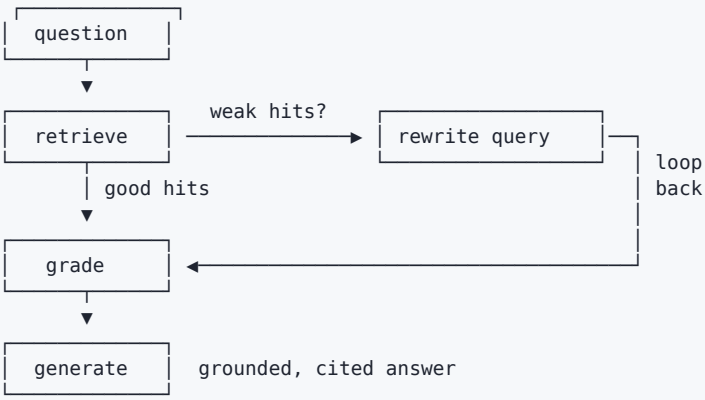
The idea

A straight RAG chain (Lessons 1 and 7) is **linear**: retrieve → ground → answer, once. Real assistants need **control flow** - decide whether to retrieve at all, retry with a reformulated query when results are weak, call more than one tool, and remember the conversation. **LangGraph** models this as a **stateful graph** of nodes and edges with explicit state.

This lesson turns the RAG pipeline into an agent that **thinks about its own retrieval**.

What you'll build

A graph with nodes such as:



What you'll learn

- **State** - a typed graph state that flows between nodes (question, docs, draft, attempts)
- **Nodes & edges** - retrieval, grading, query rewriting, and generation as composable steps

- **Conditional edges** - branch on "are these results good enough?" and loop to self-correct
- **Cycles** - corrective RAG: re-retrieve with a better query instead of answering badly
- **Memory & checkpoints** - persist state so a conversation survives across turns
- **Tool routing** - combine the Lesson 2 MCP `search_docs` tool with other tools

Builds on the whole course

Concept	Comes from
retrieve / ground / answer	Lesson 1
`search_docs` as a callable tool	Lesson 2
LangChain retrievers & chat models	Lesson 7
graph, state, cycles, self-correction	**Lesson 8 (this one)**

> **Where it all lands:** by the end of the course you can build a local, private, self-correcting document agent - and explain every layer, from `pypdf` extraction up to a LangGraph control loop.

Prerequisites

Lessons 1, 2, and 7 · LangChain. The agent reuses the retriever, the MCP tool, and the LangChain components from earlier lessons.

Next lesson

Lesson 9 · Ollama + Function Calling → - give a local model its own tools, fully offline, then carry the same `search_docs` capability through Semantic Kernel, Bedrock, and ADK.

Course: nikolareljin.github.io/local-ai-lab · **Author:** Nik Reljin