SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 02 · RAG & Knowledge Systems

09

LangChain & LlamaIndex

Build production RAG apps with top orchestration libraries

Module 09 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Build production RAG apps with top orchestration libraries. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	09/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 02

02 Learning Objectives

- Compose pipelines with LangChain Expression Language (LCEL).
- Build query engines and data connectors with LlamaIndex.
- Choose the right library (or none) for a given job.
- Stream, cache, and trace orchestrated chains.

03 Core Concepts

LCEL composition

LangChain's Expression Language pipes components — prompt | model | parser — into declarative chains that support streaming, batching, and async out of the box. It standardises how you wire retrieval, generation, and post-processing.

LlamaIndex data framework

LlamaIndex specialises in ingestion and indexing: 160+ data connectors, node parsers, and query engines (vector, summary, knowledge-graph). It's the fastest path from 'my data' to a working RAG query engine.

When to use which

Use LlamaIndex when the problem is data ingestion + retrieval; use LangChain/LangGraph when you need general orchestration and agent control flow. They interoperate — many apps use LlamaIndex for retrieval inside a LangChain pipeline. Don't over-abstract: sometimes a few API calls beat a framework.

Production concerns

Add streaming for UX, caching for cost, and tracing (LangSmith) for debugging. Keep prompts and chains versioned and covered by evals.

04 Visual Diagram

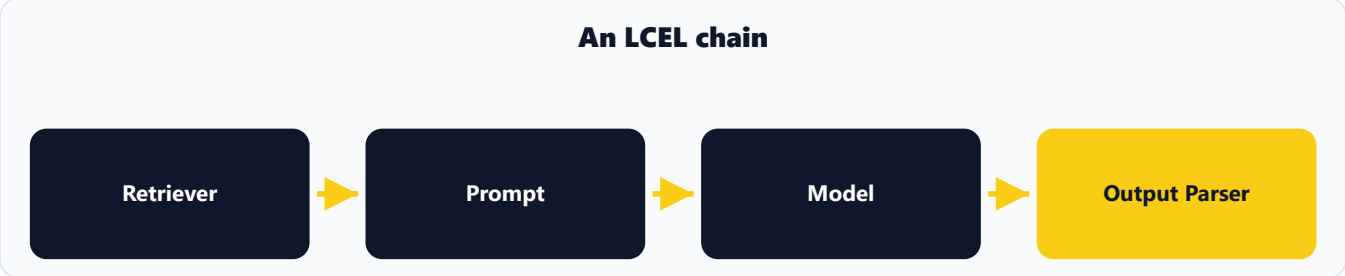

Figure 9 — LangChain Expression Language pipes components into a streaming-ready chain.

05 Key Frameworks & Tools

LangChain (LCEL) · LlamaIndex · LangSmith (tracing) · Vector DB of choice

06 Hands-On Build

🛠 Composable chains with LCEL and Query Engines

1. Build a retrieval-augmented LCEL chain: retriever | prompt | model | parser.
2. Ingest documents with LlamaIndex and expose a query engine.
3. Add streaming and a cache layer.
4. Trace a run in LangSmith and identify the slowest step.

07 Code Walkthrough

An LCEL RAG chain — PYTHON

```python
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

chain = (
    {"context": retriever, "question": RunnablePassthrough()}
    | prompt
    | model
    | StrOutputParser()
)
print(chain.invoke("What is our refund policy?"))
```

08 💉 Lab Assistant

Lab 9 · Compose an LCEL RAG chain

Prerequisites:pip install langchain langchain-core · A retriever + model

1. **Pipe retriever | prompt | model | parser**
   > Expected: chain.invoke(question) returns a grounded answer
2. **Switch invoke() to stream()**
   > Expected: Tokens arrive incrementally
3. **Wrap with a cache**
   > Expected: Repeat queries return instantly and free

Troubleshooting

- Type errors in the pipe → match each component's input/output schema.
- No streaming → ensure every component supports streaming.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Over-abstracting simple tasks behind heavy frameworks.
- Mixing framework versions with breaking API changes.
- Shipping without tracing, then being unable to debug latency.

✓ KEY TAKEAWAYS

- LCEL gives declarative, streaming-ready pipelines.
- LlamaIndex is the shortest path from data to query engine.
- Pick the lightest tool that fits; instrument with tracing.

📺 **Companion video:** Watch the module 09 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

LangChain LCEL docs • LlamaIndex docs • LangSmith tracing guide