Local RAG Pipeline with Kubernetes & Ollama

A fully local, containerized Retrieval-Augmented Generation (RAG) system built from first principles using Python, Docker, and Kubernetes. This project avoids high-level RAG abstractions in favor of explicit, inspectable system boundaries.

1. Overview

This repository implements a local-first RAG pipeline that ingests documents, builds a persistent index, retrieves relevant context, and generates grounded answers using a local LLM. All components run locally with no cloud dependencies.

2. System Architecture

High-Level Flow

```
PDF Docs
   |
Seed Docs Job
   |
Docs PVC
   |
Ingest Job (chunk + embed)
   |
Index PVC (local_index.pkl)
   |
Query Job
   |
Ollama LLM Deployment
```

Component Responsibilities

```
seed-docs     : Copy PDFs into persistent storage
rag-ingest    : Chunk documents and build index
rag-query     : Retrieve context and generate answer
ollama        : Local LLM inference service
PVCs          : Persistent documents, index, and model weights
```

3. Model & Algorithm Choices

Chunking is deterministic and transparent. Embeddings are computed locally and stored in a serialized index. Retrieval uses top-K similarity search with full visibility into scores and source metadata.

LLM Backend

Ollama is used for fully local inference. The llama3.1:8b model was selected to balance reasoning capability with local resource constraints.

4. Why Docker and Kubernetes

Docker ensures reproducible execution environments and prevents silent dependency drift. Kubernetes cleanly separates batch workloads (ingest/query) from long-lived services (LLM inference), while making storage dependencies explicit through PersistentVolumeClaims.

5. Execution Flow

```
kubectl apply -f k8s/00-namespace.yaml
kubectl apply -f k8s/01-pvc.yaml
kubectl apply -f k8s/02-ollama.yaml
kubectl apply -f k8s/03-seed-docs-job.yaml
kubectl apply -f k8s/04-job-ingest.yaml
kubectl apply -f k8s/05-job-query.yaml
```

6. Repository Structure

```
llm_project_rag_local/
■■■ data/
■■■ docs/
■■■ rag_local/
■■■ scripts/
■■■ k8s/
■■■ Dockerfile
■■■ requirements.txt
■■■ README.md
```

7. Project Rationale

This project demonstrates systems-level understanding of RAG pipelines: explicit data flow, persistence boundaries, failure modes, and separation of concerns. It favors engineering clarity over abstraction or novelty.