ALTA3 RESEARCH

Building Production RAG Systems

- 5 Days Course
- Hands-on and Lecture

Course Overview

Students build a single, production-grade repository (production-rag/) incrementally across 5 days. Hands-on lab exercises standardize on a single, enterprise-grade reference architecture covering document parsing, hybrid search, authorization, structured truth, model serving, evaluation, and observability.

Who Should Attend

- Enterprise AI Engineers and ML Developers (Suggested)
- System Architects and Infrastructure Engineers (Suggested)
- Technical Leads building custom, production-grade RAG systems (Suggested)

What You'll Learn

- Architect and maintain a production-grade RAG repository using enterprise reference standards.
- Implement Docling ingestion, metadata lineage, content hashing, and tombstone deletion workflows.
- Configure OpenSearch hybrid search, TEI reranking, and measure baseline retrieval metrics (Recall@k, MRR, NDCG).
- Integrate fine-grained authorization using OpenFGA with query pre-filtering and defend against prompt injection attacks.
- Build programmed and model-classified routing engines across heterogeneous data sources (DuckDB, OpenSearch, Crawl4AI).
- Assemble context, deploy local model serving using vLLM, and enforce structured outputs with Outlines.
- Instrument full-stack telemetry using OpenTelemetry/SigNoz, run automated evaluation with Arize Phoenix/Ragas, and validate fail-closed chaos resilience.

Outline

1. Production RAG Architecture & Trust Domains

- RAG as Information Retrieval + Context Assembly + Model Generation
- "IR is solved enough to source. Your institution isn't."
- Defining trust domains: public, institutional, and restricted/instance-specific boundaries
- Treating the vector index as derived state vs. the source system of record
- Why search_everything() is a systemic anti-pattern

2. Metadata Schemas & Corpus Lifecycle

- Metadata as the primary schema of a RAG system; embeddings as auxiliary indexes
- Required metadata fields: document_id, version, tenant_id, classification, effective_date, nda_scope
- Document lineage, content hashes, and change detection

- Tombstone, deletion, retention, and re-indexing workflows without losing provenance

3. Document Parsing & Structure Extraction

- Multi-column layout analysis, reading-order extraction, and table preservation
- Semantic chunking vs. parent-child document strategies
- Sourcing parser capabilities: Docling, Marker, Tika, Unstructured, OCRmyPDF
- Lab standard focus: Standardizing ingestion on Docling

4. Representation Infrastructure & Reranking Engines

- Decoupling generation engines from representation servers
- Sourcing representation servers: Text Embeddings Inference (TEI), Infinity
- Two-stage retrieval mechanics: candidate retrieval vs. cross-encoder reranking (e.g., BGE-Reranker)

5. Open-Source Information Retrieval (IR) Engines

- Lexical sparse search (BM25) vs. dense vector similarity
- Sourcing search engines: OpenSearch, Elasticsearch, Vespa, Weaviate, Qdrant
- Algorithmic rank fusion: Reciprocal Rank Fusion (RRF) vs. weighted score normalization

6. In-Line Retrieval Evaluation

- Measuring retrieval performance before model generation
- Baseline evaluation metrics: Recall@k, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG)
- Building small judgment sets to quantify retrieval precision shifts
- Lab 1: production-rag/ingest, metadata & retrieval
- Lab 1: Initialize the persistent production-rag/ monorepo
- Lab 1: Implement a Docling ingestion worker to parse multi-column financial and clinical PDFs containing nested tables into clean Markdown
- Lab 1: Attach explicit metadata schemas, generate version lineage hashes, and handle document deletion/tombstone events in the index
- Lab 1: Spin up OpenSearch and TEI services within the production-rag/ environment
- Lab 1: Construct a 30-query evaluation judgment set for the ingested corpus
- Lab 1: Benchmark search stages and measure metric shifts: BM25 -> Dense Vector -> Hybrid RRF -> Hybrid + Reranker

7. Identity & Fine-Grained Authorization (RBAC / ABAC / ReBAC)

- Enterprise SSO integration via OIDC, OAuth2, SAML, and Passkeys
- Sourcing identity engines: ZITADEL, Keycloak, Authentik, Ory, Authelia
- Sourcing policy engines: OpenFGA, Open Policy Agent (OPA), Cerbos, Casbin, Cedar
- Design Tension: Policy -> Search (pre-filtering via query injection) vs. Search -> Policy (post-filtering retrieved candidates)
- Injecting dynamic security pre-filters into vector query payloads

8. Corpus Trust, Poisoning & Retrieval Attacks

- Threats in RAG: direct vs. indirect prompt injection via retrieved documents
- Document poisoning, malicious metadata injection, and cross-tenant data exfiltration
- Separating retrieved untrusted data from trusted prompt instructions
- Least-privilege tool execution and sandbox boundaries
- Lab 2: production-rag/authz
- Lab 2: Configure an OpenFGA policy engine with nested relationships (User Alice -> Member -> Research Team -> Assigned -> Study X -> Permits -> Corpus Y)

- Lab 2: Intercept user tokens and inject runtime security metadata filters into OpenSearch queries
- Lab 2: Execute an indirect prompt injection attack through a poisoned PDF and implement prompt-instruction boundary defenses

9. Programmed vs. Model-Classified Routing

- Programmed routing: explicit endpoints, typed requests, known schemas, and API contracts
- Model-classified routing: natural language, LLM classifiers, structured schema outputs, and application permission validation
- When to use deterministic code vs. model classification
- Replacing heavy orchestration frameworks with thin, application-owned control

10. Structured Truth & External Web Extraction

- Querying structured databases: DuckDB, PostgreSQL, ClickHouse, Trino, Apache Drill
- Web extraction vs. metasearch: why web crawlers (Crawl4AI) beat metasearch aggregators for RAG
- Domain-specific APIs: HAPI FHIR, Medplum, Blaze FHIR
- Lab 3: production-rag/router
- Lab 3: Implement both routing strategies in production-rag/router (Programmed Branch vs Model-Classified Branch)
- Lab 3: Route structured metrics queries to DuckDB, institutional document queries to OpenSearch, and live web extraction requests to Crawl4AI
- Lab 3: Merge heterogeneous responses into a unified application data contract

11. Context Assembly & Prompt Engineering

- Context window token budgeting, trimming, and "lost-in-the-middle" re-ordering
- Preserving source span IDs and document lineage across context windows
- Structuring system prompts to enforce citation grounding and prevent post-hoc citation hallucination

12. Private Model Serving & Constrained Generation

- Sourcing GPU model serving engines: vLLM, SGLang, llama.cpp, Ollama, Hugging Face TGI
- Constrained generation: forcing structured JSON outputs using Outlines, Guidance, or grammars
- Output schema validation vs. generic LLM guardrail wrappers
- Lab 4: production-rag/context & model
- Lab 4: Deploy a local vLLM inference instance running an open-weights LLM
- Lab 4: Build a context assembly module in production-rag/context that orders snippets, manages token limits, and preserves citation source spans
- Lab 4: Force structured JSON output with precise citation spans using an Outlines constraint proxy

13. System Observability & Automated CI/CD Evaluation

- Sourcing evaluation engines: Arize Phoenix, Ragas, DeepEval, Promptfoo, Opik
- Separating retrieval metrics (Recall@k, NDCG) from generation metrics (Faithfulness, Answer Relevance)
- Full-stack distributed tracing with OpenTelemetry and SigNoz

14. Production Resilience & Failure Semantics

- Failure modes: handling outages in OpenSearch, TEI, rerankers, policy engines, and LLM servers
- Enforcing fail-closed behavior for security and policy service failures
- Implementing degraded execution modes (e.g., falling back to hybrid BM25/vector when the cross-encoder times out)
- Circuit breakers, bounded retries, rate limiting, and dead-letter queues
- Lab 5: production-rag/eval, telemetry & Chaos Testing

- Lab 5: Instrument the complete production-rag/ monorepo with OpenTelemetry and visualize traces in SigNoz
- Lab 5: Execute an automated test suite using Arize Phoenix / Ragas against a Golden Dataset to block regressions in CI/CD
- Lab 5: Capstone Chaos Exercise: Systematically kill components of the running stack and evaluate system response (TEI/Reranker, OpenFGA Policy Engine, vLLM)

Prerequisites

Next Courses