Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Retrieval-Augmented Code Generation: Grounding Agents in Your Codebase

Class Duration

14 hours of live training delivered over 2-3 days to accommodate your scheduling needs.

Student Prerequisites

- Professional software development experience in Python or TypeScript
- Basic familiarity with vector search concepts is helpful but not required

Target Audience

Software engineers and ML engineers building internal AI coding tools, developer assistants, or code search systems on top of large private codebases. Also relevant for teams extending GitHub Copilot or Claude Code with organization-specific context via MCP or RAG pipelines.

Description

Off-the-shelf AI coding assistants know open-source code but not your private codebase. This course covers the architectures and techniques for grounding AI agents in proprietary code: code-specific embedding models, AST-aware chunking strategies, hybrid semantic/keyword search, reranking, retrieval pipeline design, evaluation, incremental indexing for evolving repositories, and end-to-end integration with AI coding agents. Labs build a working code retrieval pipeline against a realistic multi-language monorepo, evaluate it against a hand-built golden set, surface it as an MCP tool to an AI coding agent, and wire up incremental re-indexing in CI. The course is the code-specific complement to **Production RAG Systems** — it goes deep on the issues unique to grounding agents in code rather than documents.

Learning Outcomes

- Explain the RAG architecture applied to code retrieval: chunking, embedding, indexing, retrieval, reranking, and augmentation, and where it differs from document RAG.
- Select and apply code-aware embedding models and AST-aware chunking strategies (tree-sitter, function-level, contextual).

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

- Build a vector index for a codebase using at least one vector database (pgvector, Qdrant, or equivalent).
- Implement hybrid search combining dense embedding retrieval with BM25 keyword search and tune Reciprocal Rank Fusion weights.
- Apply a reranking model to improve retrieval precision for code queries.
- Build a hand-curated golden dataset and evaluate retrieval quality using Recall@K, Precision@K, MRR, and nDCG.
- Diagnose retrieval failure modes and apply targeted fixes (chunk boundaries, query rewriting, metadata filters).
- Operate a retrieval pipeline at production quality: incremental re-indexing in CI, embedding model migrations, and observability.
- Expose a code retrieval pipeline as an MCP tool for Claude Code or GitHub Copilot and validate the end-to-end agent-grounded workflow.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

Python 3.12+, Docker (for running local vector DBs — pgvector and Qdrant), Git, and the sample multi-language monorepo provided for labs. Embedding and rerank API access (Voyage, Cohere, or OpenAI) is provided for the duration of the course.

Training Topics

RAG for Code: Architecture Overview

- Why plain context windows aren’t enough for large codebases
- RAG pipeline stages: chunk, embed, index, retrieve, rerank, augment
- Code-specific challenges vs. document RAG: symbols, cross-file references, naming, and code drift
- Reference architectures and where code-RAG fits in agent workflows

Code Chunking Strategies (with hands-on)

- Naive chunking: line-based and token-based, and why they fail for code
- AST-aware chunking with tree-sitter across multiple languages

- Function, class, and module-level chunks
- Late chunking and Contextual Retrieval applied to code (per-chunk context summaries for cross-file references)
- Cross-file dependency context preservation: imports, callers, type references
- Hands-on lab: implement two chunking strategies on the sample monorepo and compare retrieval quality

Embedding Models for Code

- General-purpose vs. code-specific embedding models
- Voyage Code, OpenAI text-embedding-3, Cohere for code, and current open-source options
- Multi-language embedding considerations
- Embedding quality evaluation and how to choose
- Hands-on lab: benchmark two embedding models against a small golden set

Vector Indexing

- pgvector, Qdrant, Chroma, and Pinecone — concrete tradeoffs
- HNSW index configuration, ef_search/ef_construction tuning, and recall tradeoffs
- Metadata fields for code: file path, language, symbol, version, last-modified
- Hands-on lab: stand up pgvector and Qdrant indexes for the same dataset and compare

Hybrid Search

- BM25 keyword search alongside dense retrieval
- Reciprocal Rank Fusion (RRF) for result merging and weight tuning
- When hybrid search outperforms dense-only — and when it doesn’t
- Query rewriting and symbol-aware query expansion
- Hands-on lab: enable hybrid search and tune RRF weights against the golden set

Reranking

- Cross-encoder rerankers for code retrieval
- Cohere Rerank, Voyage Rerank, and local alternatives
- Latency vs. quality tradeoffs and when reranking pays for itself
- Hands-on lab: add a reranker and measure quality and latency impact

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Retrieval Pipeline Evaluation

- Recall@K, Precision@K, MRR, and nDCG — what each measures and when to use it
- Building a golden dataset for code retrieval evals (queries, expected files, expected symbols)
- LLM-assisted golden-set construction and validation
- Failure-mode analysis: what the retriever misses and why
- Regression testing for retrieval quality across changes
- Hands-on lab: build a golden set and run a full eval suite

Production Operations

- Incremental indexing for evolving codebases (commit-level, branch-aware)
- Re-indexing in CI: cost, latency, and freshness tradeoffs
- Embedding model migrations and dual-index strategies
- Observability: query latency, hit rate, retrieval-to-generation correlation
- Cost controls: caching, embedding budgets, and tier-based retrieval
- Hands-on lab: wire up incremental re-indexing on commit

Exposing Retrieval as an AI Tool

- MCP tool wrapping a retrieval pipeline
- Context packaging: file path, symbol, snippet metadata, and ranking signals
- Truncation, chunk merging, and budget-aware result selection
- Integrating with Claude Code and GitHub Copilot
- Validating end-to-end: does grounded retrieval improve agent task performance?
- Hands-on lab: ship the retrieval pipeline as an MCP tool and use it from Claude Code

Capstone Workshop

- Build, evaluate, and integrate a complete code retrieval pipeline against the sample monorepo
- Present results: chosen architecture, eval scores, and end-to-end agent demo
- Group critique and Q&A

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

- Recommended follow-on path: **Production RAG Systems** for general document RAG depth, **Building Custom Tools and Skills** for richer MCP tool design