A Practical Guide to Building Production-Grade RAG Systems

Author: Kamakshaiah Musunuru | **Published:** 11 September 2026 | **Category:** Blog | **Read time:** 2 min read

Tags: RAG, generative AI, LLM, retrieval-augmented generation, vector databases, embeddings

Retrieval-augmented generation (RAG) grounds an LLM in your own data: instead of asking the model to answer from memory, we retrieve relevant passages from a knowledge base and let the model answer from those passages. Done well, RAG gives you accurate, attributable and updatable answers.

The Core Pipeline

1. Ingestion and Chunking Documents are cleaned and split into chunks sized for retrieval. Good chunking respects structure — split on headings and paragraphs rather than fixed character counts — and keeps each chunk self-contained.

2. Embeddings Each chunk is converted into a vector embedding that captures semantic meaning. The choice of embedding model affects quality and cost, so evaluate it against your domain vocabulary.

3. Vector Search User queries are embedded and matched against the chunk store (pgvector, Pinecone, Weaviate, Qdrant, or similar) using similarity search. Hybrid search — combining vector similarity with keyword (BM25) matching — improves recall for exact terms like product codes or policy numbers.

4. Reranking The top matches from vector search are re-scored by a reranking model against the actual question. This single step typically delivers the largest quality improvement for the least cost.

5. Generation with Grounding The retrieved passages and the user question are composed into a prompt with clear instructions: answer only from the context, cite sources, and say when the context is insufficient. The model never sees unrelated data.

Guardrails for Production

- Prompt injection protection: instruct the model to ignore instructions found in retrieved text and treat retrieved content as data, never as commands - PII redaction before retrieval and before generation - Output validation and content moderation on generated answers - Hallucination controls: require the answer to be supported by a retrieved source - Evaluation: build a golden dataset of question-answer pairs and track retrieval quality (recall@k), answer accuracy and citation fidelity across versions

Operating a RAG System

Embeddings and index updates must stay in sync with your source documents. Monitor retrieval quality over time with analytics — low-quality answers usually trace back to poor chunking or missing documents, not to the model.

Codingfigs designs and deploys RAG systems with evaluation frameworks, guardrails and human-in-the-loop review built in. Contact us to discuss your knowledge base and use case.

Article Statistics

43	24	0
Views	Downloads	Citations

Read online: https://codingfigs.com/knowledge/a-practical-guide-to-building-production-grade-rag-systems/

© 2026 Codingfigs. All rights reserved. Reproduction without permission is prohibited.