FREE PLAYBOOK · AI TRAINING SERIES · 2026 EDITION

RAG Implementation Playbook

The complete guide to building production RAG — chunking, embeddings, vector databases, evaluation.

Hareekrishna Ravichandran
Founder, Hari Advisory
AI Trainer · Supply Chain Consultant
consult@hariadvisory.com
12+ years · Blinkit · Zepto · Flipkart · HP
+91 98801 27460

Introduction

Retrieval-Augmented Generation — RAG — is how most enterprises are actually using LLMs today. Not chatbots. Not agents. RAG. Because RAG solves the one thing LLMs cannot do on their own: reason over your specific knowledge, documents, and data.

This playbook is the same one I use with corporate training clients. It covers every stage of building a production RAG system, the common failure modes, and how to evaluate whether it is actually working.

> *"RAG works by combining retrieval — finding the right information from your data — with generation — using an LLM to answer questions based on that information."*

Part 1: RAG Architecture Fundamentals

The RAG Pipeline

Every RAG system has the same core components:

- **Ingestion** — Load your documents (PDFs, docs, wikis, database records)
- **Chunking** — Split them into retrievable pieces
- **Embedding** — Convert each chunk into a numerical vector
- **Storage** — Store vectors in a specialised vector database
- **Retrieval** — At query time, find the most relevant chunks
- **Generation** — Pass those chunks to an LLM to generate the answer

Part 2: Chunking Strategy

Chunking is where most RAG systems fail silently. Choose the wrong chunk size or strategy, and your retrieval will consistently miss the right information. There are four strategies I use in practice:

STRATEGY	WHEN TO USE	CHUNK SIZE
Fixed-size	Simple text, uniform structure	500-1000 tokens
Semantic	Long documents with clear sections	Variable
Sentence-based	Q&A style content, FAQs	2-5 sentences
Hierarchical	Complex documents with sections	Parent + child chunks

> *Rule of thumb: start with 500-token chunks and 100-token overlap. Iterate from there based on evaluation results.*

Part 3: Choosing Your Vector Database

The vector database is where your embeddings live. Choose wrong, and you either burn money or hit scaling limits. Here is my current recommendation matrix:

DATABASE	BEST FOR	COST
Chroma	Prototyping, small deployments	Free / self-hosted
Pinecone	Managed, easy scaling	Paid, usage-based
Weaviate	Hybrid search, complex queries	Free / paid tiers
Qdrant	High performance, self-hosted	Free / self-hosted
pgvector (Postgres)	Already using Postgres	Free

Part 4: Embedding Models

The embedding model is what turns text into vectors. The quality of your embeddings directly determines retrieval quality. Do not skimp here.

- **OpenAI text-embedding-3-small** — Cheap, fast, works for most cases
- **OpenAI text-embedding-3-large** — Best quality, higher cost
- **Cohere embed-v3** — Multi-lingual, good for Indian languages
- **BGE (open-source)** — Free, self-hosted, competitive quality

Part 5: Advanced Retrieval Techniques

Hybrid Search

Pure vector search misses exact matches (like product codes, IDs, names). Hybrid search combines vector similarity with traditional keyword search (BM25). This is the single biggest quality boost you can add to a production RAG system.

Re-ranking

After initial retrieval, use a cross-encoder to re-rank the top results. This adds latency but improves relevance significantly. Cohere Rerank is the easiest option; open-source alternatives include bge-reranker.

Query Expansion

Use an LLM to expand the user's question into multiple related queries, then retrieve for each. This helps when users ask short or ambiguous questions.

Part 6: Evaluation

You cannot improve what you cannot measure. Every production RAG system needs an evaluation harness. Measure three things:

- **Retrieval quality** — Are we finding the right chunks? (Recall@K, MRR)
- **Answer quality** — Is the LLM generating correct answers? (Faithfulness, relevance)
- **User satisfaction** — Are users actually using it? (Feedback loops, session length)

> *Ragas and TruLens are the two open-source evaluation frameworks I recommend to clients.*

Part 7: Production Deployment Checklist

- Rate limiting and cost caps in place
- Fallback logic when retrieval finds no relevant chunks
- Confidence scores exposed to users

- Citation of source documents in every answer
- Monitoring for hallucination rates
- User feedback capture (thumbs up/down)
- Regular re-embedding as documents change
- Access control aligned with source documents

Next Steps

This playbook covers the fundamentals. To go deeper:

- Enrol in the Hari Advisory **RAG Training Workshop** — 3-week cohort
- Book a discovery call to discuss your specific documents and use case
- Read the AI Agent Starter Guide for building agents on top of RAG

READY TO GO FURTHER?

Book a Free Discovery Call

A 30-minute session to discuss your operations, AI adoption plans, or team training needs. No obligation. No sales pitch. Just useful conversation.

EMAIL	consult@hariadvisory.com
PHONE	+91 98801 27460
LINKEDIN	linkedin.com/in/hareekrishna-r
WEBSITE	hariadvisory.com