SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 02 · RAG & Knowledge Systems

10

Production RAG Systems

Evaluation, monitoring & optimising RAG at scale

Module 10 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Evaluation, monitoring & optimising RAG at scale. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	10/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 02

02 Learning Objectives

- Evaluate RAG with faithfulness, relevance, and context metrics.
- Build an automated eval suite with RAGAS.
- Monitor cost, latency, and quality in production.
- Optimise the retrieval/generation trade-off.

03 Core Concepts

RAG-specific metrics

Beyond 'is the answer good', measure faithfulness (is it grounded in context?), answer relevance, context precision (did we retrieve the right chunks?) and context recall. These isolate whether failures come from retrieval or generation.

Automated evaluation

RAGAS and LLM-as-judge pipelines score these metrics on a labelled set automatically, so you can compare config changes objectively and catch regressions in CI rather than in production.

Observability

Track per-request cost, latency at each stage, retrieval hit rate, and user feedback. Dashboards surface drift — e.g. a new document type tanking retrieval — before users complain.

Optimisation levers

Tune chunk size, top-k, reranking, model choice, and caching against cost/latency/quality. Often a smaller model + better retrieval beats a bigger model + weak retrieval at a fraction of the cost.

04 Visual Diagram

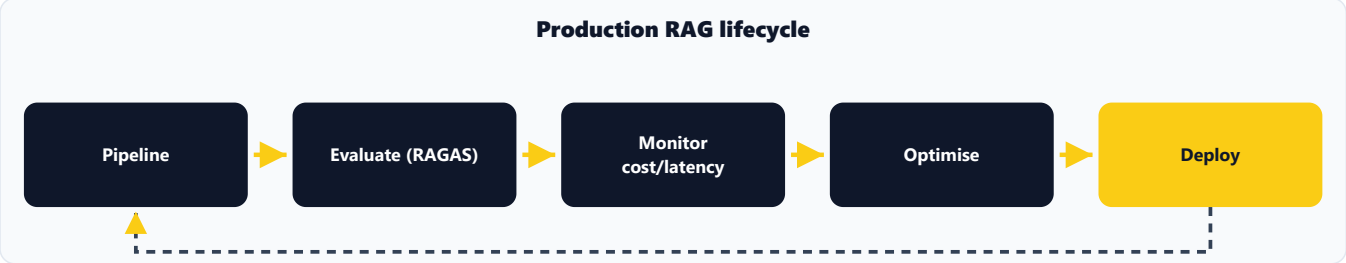

Figure 10 — RAG is never 'done': evaluate, monitor, and optimise continuously.

05 Formula Spotlight

Context precision

$$\text{precision} = |\ \text{relevant} \cap \text{retrieved}\ | \ / \ |\ \text{retrieved}\ |$$

`retrieved`	the set of chunks the retriever returned
`relevant`	the chunks that actually help answer the query
`∩`	intersection — relevant AND retrieved

Context precision isolates retrieval quality: of everything you retrieved, what fraction was useful? A low score means you are stuffing the prompt with noise — fix retrieval before blaming the model.

06 Key Frameworks & Tools

RAGAS · LangSmith / Phoenix (Arize) · Prometheus + Grafana · LLM-as-judge

07 Hands-On Build

🛠 RAGAS Evaluation suite + Cost/Latency Dashboards

1. Create a labelled eval set of question/ground-truth pairs.
2. Run RAGAS for faithfulness, relevance, context precision/recall.
3. Instrument the pipeline to emit cost + latency per stage.
4. Build a dashboard and run an A/B on chunk size or top-k.

08 Code Walkthrough

Evaluating a RAG pipeline with RAGAS — PYTHON

```python
from ragas import evaluate
from ragas.metrics import faithfulness, answer_relevancy, context_precision

result = evaluate(
    dataset,                      # questions, answers, contexts, ground_truths
    metrics=[faithfulness, answer_relevancy, context_precision],
)
print(result)  # per-metric scores you can track over time
```

09 🧪 Lab Assistant

Lab 10 · Evaluate RAG with RAGAS

Prerequisites:pip install ragas datasets · A labelled question/ground-truth set

1. **Assemble a dataset (questions, answers, contexts, ground_truths)**
   > Expected: A datasets object RAGAS accepts
2. **Run faithfulness, answer_relevancy, context_precision**
   > Expected: Per-metric scores 0…1
3. **Change chunk size and re-evaluate**
   > Expected: Scores move — pick the better config

Troubleshooting

- All scores low → check contexts actually contain the answer.
- Judge errors → ensure the eval model has API access.

10 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Judging RAG only by eyeballing a few answers.
- No labelled eval set, so every change is a guess.
- Optimising quality while ignoring cost and latency budgets.

✓ KEY TAKEAWAYS

- Separate retrieval vs generation failures with targeted metrics.
- Automate evals (RAGAS) to catch regressions before users do.
- Better retrieval often beats a bigger, costlier model.

📺 **Companion video:** Watch the module 10 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

11 Further Resources

RAGAS docs • Arize Phoenix • LangSmith evaluation guide