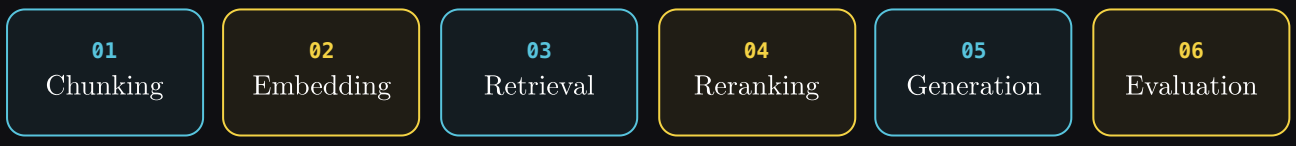
RAG INTERVIEW PREP

The Complete RAG Roadmap

The 6 steps of a retrieval-augmented generation pipeline, the concepts to know at each one, and the questions interviewers ask about them.



6 steps
The full pipeline, in order

30 core concepts
One line each

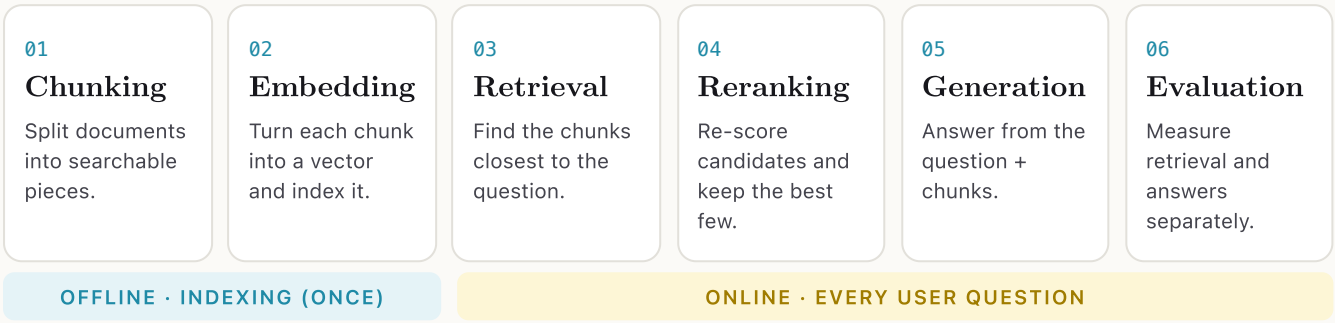
24 questions
What interviewers ask

6-week plan
One step a week

START HERE

What RAG is, in one picture

Retrieval-augmented generation pairs an LLM with an external knowledge source. Instead of answering only from what it memorised in training, the model first **retrieves** relevant text and then **generates** an answer grounded in it. This gives you up-to-date, private and citable knowledge without retraining the model.



The one rule interviewers test

When a RAG answer is wrong, first decide **where** it failed. Was the right chunk never retrieved (steps 1–4), or was it retrieved and then ignored or misused (step 5)? Evaluating those two separately (step 6) is how you debug.

CONTENTS

01 · Chunking	size, overlap, recursive, semantic, metadata	3
02 · Embedding	bi-encoders, similarity, vector stores, HNSW	4
03 · Retrieval	top-k, dense vs BM25, hybrid + RRF, HyDE	5
04 · Reranking	two-stage retrieval, cross-encoders, LLM rerank, top-n, compression	6
05 · Generation	prompt, context budget, lost in the middle, citations	7
06 · Evaluation	golden set, context recall / precision, faithfulness, LLM-as-judge	8
Cheat sheet + plan	debug a wrong answer, 6-week plan	9

STEP 1 OF 6

Chunking

WHY IT MATTERS

Retrieval can only return what chunking created. Chunks that are too big blur several topics into one vector; chunks that are too small lose the context needed to answer.

LEARN

Chunk size

How much text goes in each piece, usually counted in tokens. The main trade-off: precision vs context.

Overlap

Text repeated between neighbouring chunks, so a fact cut at a boundary still appears whole somewhere. Costs extra storage.

Recursive splitting

Split on paragraphs, then sentences, then words, until each piece fits the size. LangChain's default splitter.

Semantic splitting

Place breakpoints where the meaning changes, measured by embedding similarity between sentences (LlamaIndex `SemanticSplitterNodeParser`).

Metadata

Source, title, section, date and access rights attached to each chunk, for filtering and citations.

INTERVIEW QUESTIONS

- Q1 How do you choose chunk size and overlap?
- Q2 What breaks when chunks are too small? Too big?
- Q3 When is semantic chunking worth the extra cost?
- Q4 How would you chunk tables, code and scanned PDFs?

COMMON MISTAKES

- ✕ One chunk size for every document type.
- ✕ Splitting mid-sentence or through a table.
- ✕ Throwing away metadata, so you can't filter or cite.

Start here: recursive splitting with a small overlap, then tune chunk size against your own evaluation set (step 6), not by feel.

STEP 2 OF 6

Embedding

WHY IT MATTERS

Embeddings decide what "similar" means. If the model puts the question and the right chunk far apart, no retriever can find it.

LEARN

Embeddings

Dense vectors that place texts with similar meaning close together, so search works on meaning, not exact words.

Bi-encoder

The query and each chunk are encoded separately, so chunk vectors can be computed once, ahead of time. Fast, but less precise than a cross-encoder.

Similarity

Cosine similarity or dot product between vectors. Use the one the model was trained with.

Vector store

A database that stores vectors with their text and metadata, and answers nearest-neighbour queries.

Indexing (ANN, HNSW)

Approximate nearest-neighbour indexes trade a little accuracy for big speed. HNSW is a layered graph you walk from coarse to fine.

INTERVIEW QUESTIONS

Q1 Why can't you mix vectors from two embedding models?

Q2 How would you pick an embedding model for your domain?

Q3 What does HNSW trade off, and which knobs control it?

Q4 How do you enforce document permissions in retrieval?

COMMON MISTAKES

- ✕ Picking the top leaderboard model without testing on your data.
- ✕ Changing the embedding model without re-indexing everything.
- ✕ Ignoring permissions, so users can retrieve documents they shouldn't see.

> **Remember:** the query and the chunks must be embedded by the same model. A new model means a new index.

STEP 3 OF 6

Retrieval

WHY IT MATTERS

If the right chunk isn't in the retrieved set, the LLM can't use it. Most "the model got it wrong" bugs start here.

LEARN

Top-k

How many chunks to fetch. Too few misses answers; too many adds noise, latency and cost.

Dense (vector) search

Matches meaning and paraphrases. Weak on exact IDs, codes, names and rare terms.

BM25 (sparse search)

Classic keyword ranking. Strong on exact terms, blind to synonyms.

Hybrid search + RRF

Run both and merge with reciprocal rank fusion: each list adds 1 / (k + rank), with k usually 60. LangChain's `EnsembleRetriever` does this.

HyDE

The LLM writes a hypothetical answer, and you search with its embedding instead of the question's.

INTERVIEW QUESTIONS

Q1 Why does vector search miss an exact product code?

Q2 How do you combine BM25 and vector results?

Q3 How would you choose k?

Q4 What is HyDE, and when does it help?

COMMON MISTAKES

- ✕ Vector-only search on data full of IDs and acronyms.
- ✕ Never logging which chunks were retrieved per question.
- ✕ Adding score values from BM25 and vectors directly (different scales); fuse ranks instead.

Data point: in Anthropic's tests, contextual embeddings + contextual BM25 cut the top-20 retrieval failure rate by 49% (5.7% → 2.9%).

STEP 4 OF 6

Reranking

WHY IT MATTERS

Fast retrieval casts a wide net; reranking reads each candidate properly and keeps only the best few for the LLM.

LEARN

Two-stage retrieval

Retrieve wide (e.g. top-20 with a fast retriever), then rerank down to top-n for the prompt.

Cross-encoder

Reads the query and a chunk together and outputs one relevance score. More accurate than a bi-encoder, but one model call per pair.

LLM reranking

Ask an LLM to judge or order candidates (LlamaIndex `LLMRerank`, RankGPT). Flexible, slower and costlier.

Top-n

How many reranked chunks you keep (LlamaIndex `top_n`).

Contextual compression

Wrap a retriever so results are filtered or trimmed before generation (LangChain `ContextualCompressionRetriever`).

INTERVIEW QUESTIONS

- Q1 Bi-encoder vs cross-encoder: what's the difference?
- Q2 Why not run the cross-encoder over the whole corpus?
- Q3 How do you choose k (retrieve) and n (keep)?
- Q4 When is reranking not worth the latency?

COMMON MISTAKES

- ✕ Reranking too few candidates, so the right chunk was never there.
- ✕ Ignoring the added latency on every request.
- ✕ Adding a reranker without measuring before and after.

Data point: adding reranking on top of contextual retrieval took Anthropic's failure reduction to 67% (5.7% → 1.9%).

STEP 5 OF 6

Generation

WHY IT MATTERS

Even with the right chunks, the model can ignore them, mix them up or make things up. The prompt and the context layout decide what it actually uses.

LEARN

Prompt template

Instructions + retrieved context + the user's question, with a clear rule: answer only from the context.

Context budget

More chunks isn't better. Every chunk costs tokens and adds noise.

Lost in the middle

Models use information best at the start or end of a long context and worst in the middle. Put the strongest chunks first.

Citations

Ask the model to cite chunk IDs, so answers can be checked and trusted.

Saying "I don't know"

Tell the model what to do when the context doesn't contain the answer.

INTERVIEW QUESTIONS

Q1 The right chunk was retrieved but the answer is wrong. Why?

Q2 How do you order chunks in the prompt?

Q3 How do you make the model cite its sources?

Q4 What should happen when the answer isn't in the context?

COMMON MISTAKES

- ✕ Stuffing in as many chunks as fit.
- ✕ No instruction to stay within the context.
- ✕ No citations, so nobody can verify the answer.

> **Interview line:** "Retrieval decides what the model *can* know; the prompt decides what it *does* use."

STEP 6 OF 6

Evaluation

WHY IT MATTERS

Every choice in steps 1–5 is a guess until you measure it. Good evaluation tells you which step failed, not just that the answer was bad.

LEARN

Golden dataset

Real questions with expected answers and the chunks that contain them. Your regression test for every change.

Context recall

Did retrieval find the chunks needed to answer? Low recall → fix chunking, embeddings or k.

Context precision

How much of what was retrieved is actually relevant? Low precision → rerank or tighten filters.

Faithfulness / groundedness

Is every claim in the answer supported by the retrieved context? The hallucination check.

LLM-as-judge

An LLM scores answers against these criteria at scale. Spot-check it against human ratings.

INTERVIEW QUESTIONS

Q1 How do you evaluate a RAG system with no labels?

Q2 Faithfulness vs correctness: what's the difference?

Q3 How do you tell if retrieval or generation failed?

Q4 How would you build the first evaluation set?

COMMON MISTAKES

- ✕ One end-to-end score that hides which step broke.
- ✕ Trusting an LLM judge without checking it.
- ✕ Changing chunking or models with no regression run.

Also track in production: latency, cost per answer and user feedback (thumbs up/down, "no answer" rate).

CHEAT SHEET

Debug a wrong RAG answer

The most common RAG interview scenario: "Your chatbot gave a wrong answer. What do you check?" Walk the pipeline in order.

SYMPTOM	LIKELY STEP	WHAT TO TRY
The answer exists in the docs, but the right chunk is never retrieved	Chunking · Embedding · Retrieval	Check chunk boundaries, add hybrid search for exact terms, add query rewriting, raise k.
Right chunk is retrieved, but ranked too low to make the prompt	Reranking	Add a cross-encoder reranker; retrieve wider, keep top-n.
Right chunk is in the prompt, answer still wrong	Generation	Move the best chunk first, tighten instructions, cut noisy chunks, require citations.
Answer is fluent but not in any chunk	Generation	Faithfulness check, "answer only from context" rule, allow "I don't know".
Answers were right last month, now wrong	Data / index	Stale or missing documents; re-index, add date metadata and filters.
You can't tell which of the above it is	Evaluation	Build a golden set; measure context recall and faithfulness separately.

A 6-WEEK PLAN: ONE STEP A WEEK

Week 1 · Chunking
Index one document set three ways; compare answers.

Week 2 · Embedding
Try two embedding models; check MTEB vs your data.

Week 3 · Retrieval
Add BM25 + RRF; test exact-term queries.

Week 4 · Reranking
Add a cross-encoder; measure quality and latency.

Week 5 · Generation
Add citations and an "I don't know" rule.

Week 6 · Evaluation
Build 30 golden questions; track recall + faithfulness.

Save this. Share it with someone preparing. @datawithsuman

Data points: Anthropic, Introducing Contextual Retrieval (2024), for their test setup. Library class names from the LangChain and LlamaIndex docs, October 2026.