RAZA JAHANGIR
AI SYSTEMS ENGINEER

razajahangir.dev

AI Systems Engineering

A visual desk reference

LLMs, RAG, agents and LLMOps. One diagram per page. Only what you need to recall how it works.

Build · Evaluate · Deploy · Monitor

EDITION 1 · NOV 2026

Contents

One concept per page. Read the diagram first.

PART 1 · FOUNDATIONS

- How an LLM generates text 3
- Attention, up close 4
- Embeddings & vector search 5
- Inference: prefill, decode, KV cache 6
- How models are trained 7
- Fine-tuning with LoRA 8
- Choosing a model 9

PART 2 · RAG

- RAG: ingestion 10
- RAG: retrieval 11
- RAG: patterns 12
- RAG: evaluation 13

PART 3 · AGENTS

- The ReAct agent loop 14
- LangGraph in one page 15
- Workflows vs agents 16
- Subagents, handoffs, skills, router 17
- How agents act and plan 18
- Memory & MCP 19
- Context engineering 20

PART 4 · LLMOPS

- The LLMOps lifecycle 21
- Structured output 22
- Tracing & evals 23
- Guardrails & privacy 24
- Cost & latency 25

PART 5 · DECIDE & DESIGN

- Decision trees 26
- Numbers to know 27
- Anti-patterns 28
- Reference: enterprise RAG chatbot 29
- Reference: private AI stack 30

REFERENCE

- Formula cheat sheet 31
- Glossary A–Z 32

HOW TO READ THE DIAGRAMS

Main step · Supporting step · Data or output · Input or neutral

Masked or disabled · Optional step

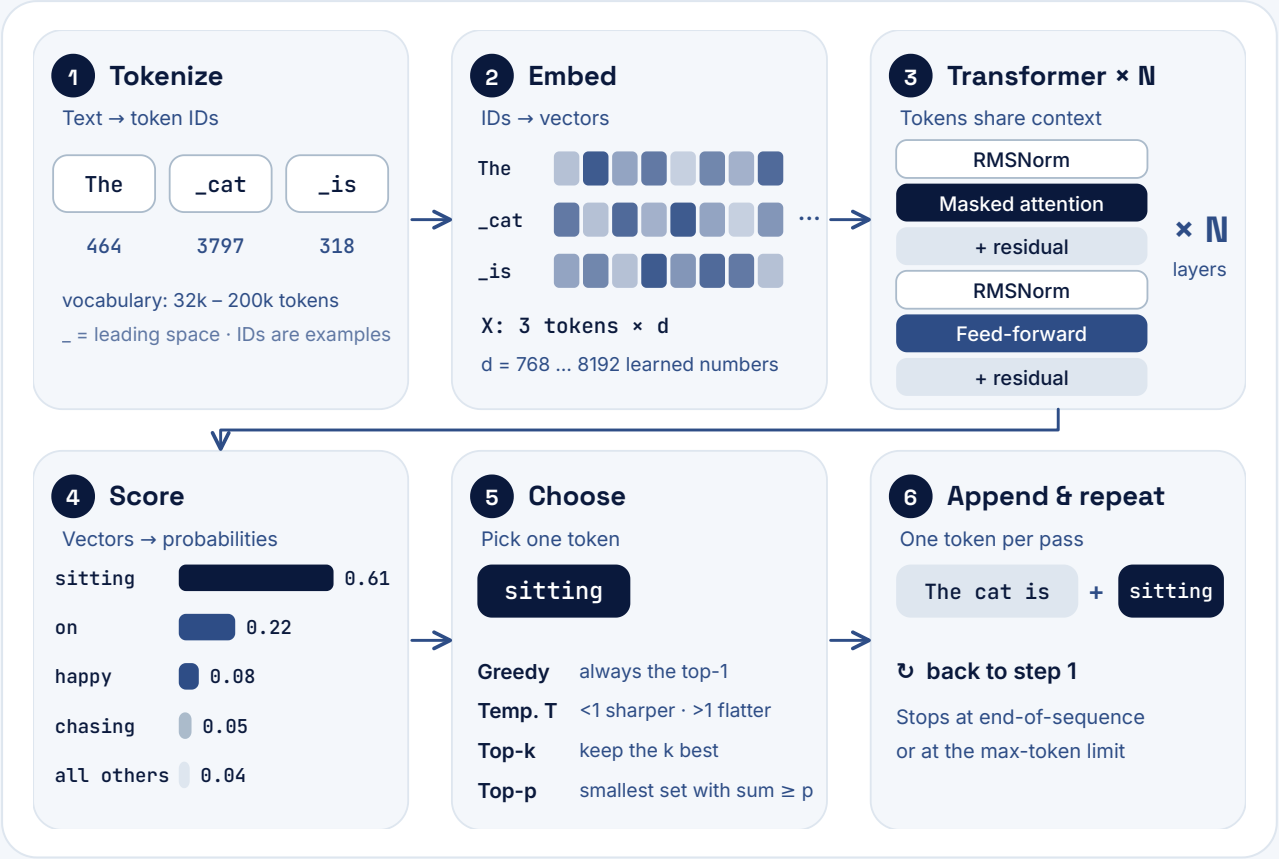
Unfamiliar word? The Glossary A–Z defines every term used. Numbers like 0.61 or 464 are examples. Dollar figures use example prices. Check your own model and provider.

SCOPE

How today's transformer models work, how to fine-tune them, and how to build RAG and agent systems on top. It is not a course in the deeper ML maths and science.

How an LLM generates text

A decoder-only transformer, one token at a time.



KEY POINTS

- An LLM is a **next-token predictor**, run in a loop.
- Only the **last position** is used to choose the next token.
- **Attention** moves information between tokens; the **feed-forward** net works on each token alone.
- Modern (Llama-style) blocks use **RMSNorm** and **SwiGLU**; GPT-2 used LayerNorm and GELU.

FORMULA

$$p(\text{next}) = \text{softmax}(h_{last} \cdot W_{vocab} / T)$$

h_{last} = last token's vector
W_{vocab} = d × vocab matrix
T = temperature

IN PRODUCTION

Prefill
Reads the whole prompt in parallel.

Decode
Then one new token per pass.

KV cache
Keeps K and V of past tokens, so each pass computes only the newest.

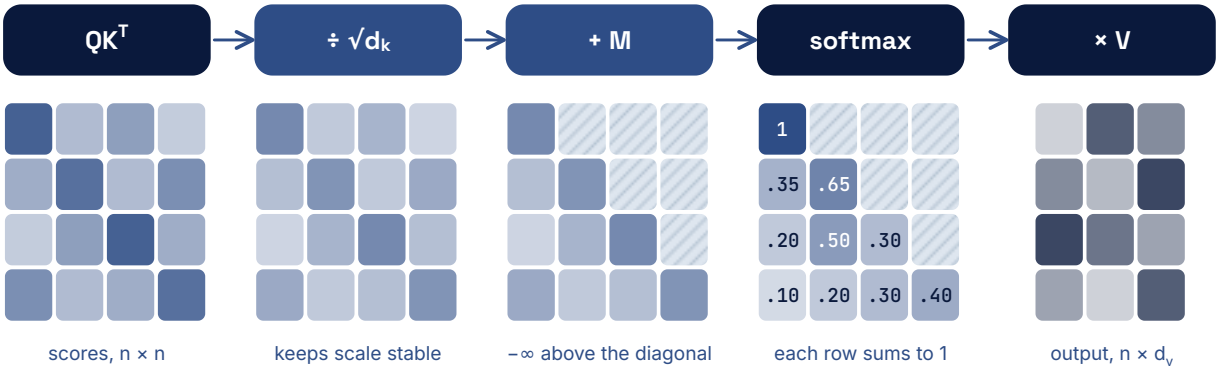
WEB-DEV LENS Like a request loop: each response is appended to the input until a stop condition fires.

Attention, up close

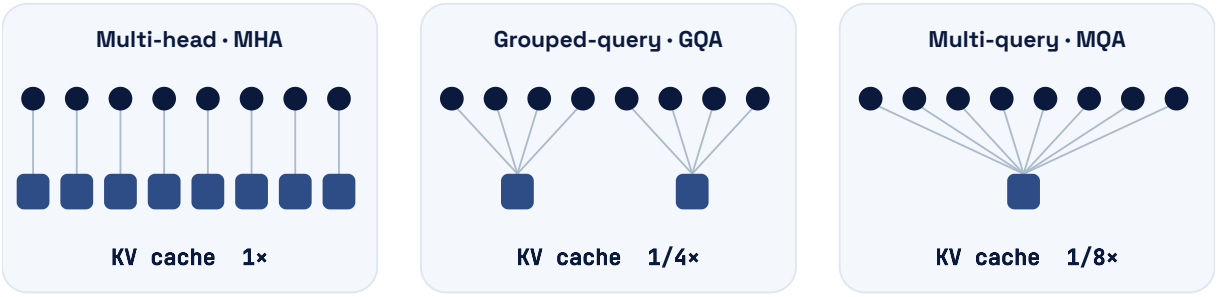
Every token looks at every earlier token and takes a weighted mix.

$$X\ (n \times d) \rightarrow Q = X \cdot W_Q \ \cdot\ K = X \cdot W_K \ \cdot\ V = X \cdot W_V$$

ONE HEAD · FIVE STEPS · each grid: rows = tokens looking, columns = tokens looked at



THREE WAYS TO SHARE K AND V ACROSS HEADS



KV CACHE SIZE

$$2 \cdot L \cdot H_{kv} \cdot d_{head} \cdot n \cdot \text{bytes}$$

L layers, H_{kv} KV heads, n tokens

$$32 \cdot 8 \cdot 128 \cdot 8{,}192 \cdot 2\ \text{B (fp16)} \approx 1\ \text{GiB per sequence}$$

KEY POINTS

- **Q** query: what this token looks for. **K** key: what a token offers. **V** value: what it passes on.
- Compute grows with **n²**; cache grows with **n**.
- **GQA** keeps most of MHA's quality with a far smaller cache.

IN PRODUCTION

Long context
Big KV cache means fewer concurrent users.

FlashAttention
Same result, less memory traffic.

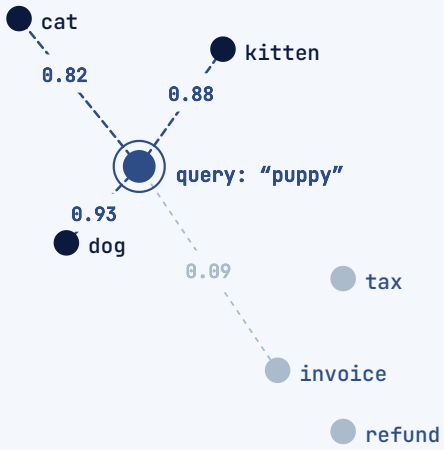
Sliding window
Trades reach for lower cost.

WEB-DEV LENS Like a JOIN with weights: each row looks up all others and blends what it finds.

Embeddings & vector search

Text becomes a point; similar meaning means a nearby point.

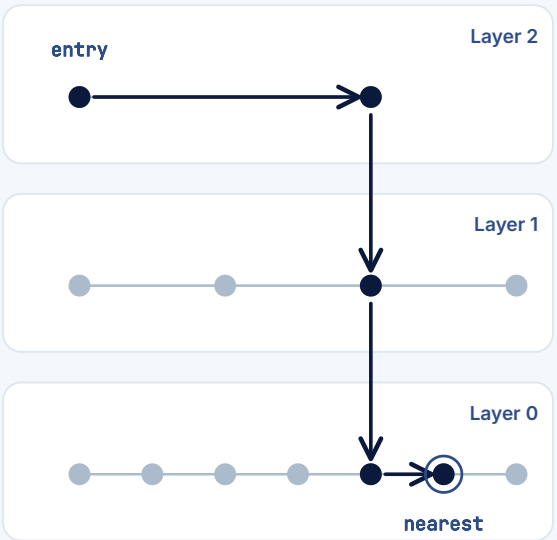
MEANING AS DISTANCE



Closer = more similar. Numbers are cosine.

2-D sketch of a d-dimensional space · example values

HNSW: FAST APPROXIMATE SEARCH



Top layers: long jumps · bottom layer: fine search

`ef_search ↑ → recall ↑, latency ↑`

SIMILARITY

$$\cos(a, b) = a \cdot b \,/\, (|a| \cdot |b|)$$

unit-length vectors: $\cos = a \cdot b$

KEY POINTS

- Typical sizes: **384 – 4,096** dimensions. Matryoshka models can be truncated.
- Use the **same model** for documents and queries.
- Change the model → **re-embed everything**.

IN PRODUCTION

Cosine
Default for text. Ignores vector length.

Dot product
Same ranking if vectors are normalised.

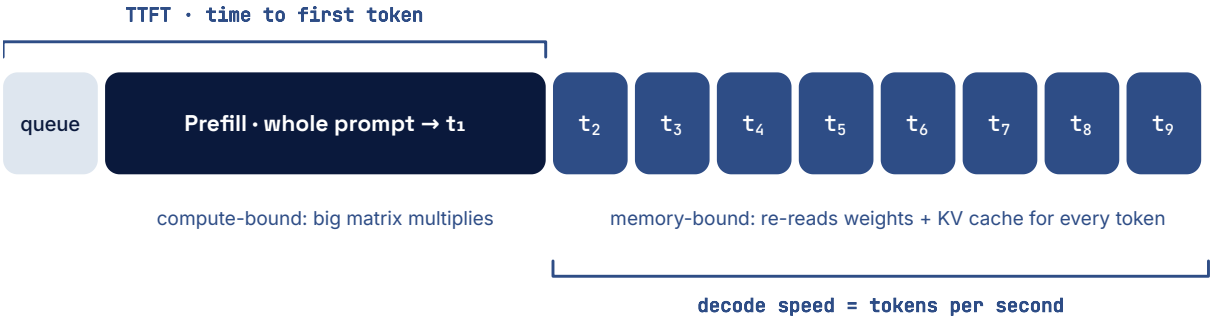
ANN index
HNSW trades a little recall for big speed.

WEB-DEV LENS Like a full-text index that matches meaning instead of letters.

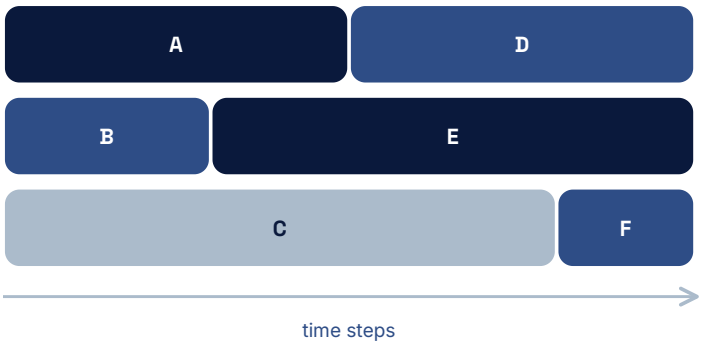
Inference: prefill, decode, KV cache

Why the first token is slow and the rest are fast.

ONE REQUEST



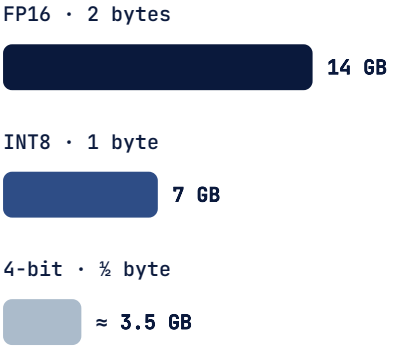
CONTINUOUS BATCHING



A finished request frees its slot at the next step.

New requests join immediately; the GPU never waits for the slowest one.

7B MODEL · WEIGHTS ONLY



KV cache and overhead come on top

MEMORY

$$\text{weights} \approx \text{params} \times \text{bytes per param}$$

$$KV = 2 \cdot L \cdot H_{kv} \cdot d_{head} \cdot n \cdot \text{bytes}$$

KEY POINTS

- **Quantization** shrinks weights; small quality loss at 8-bit, more at 4-bit.
- **PagedAttention** stores the KV cache in fixed blocks, like OS memory pages.

IN PRODUCTION

Ollama
Local and dev. Simple.

vLLM
Multi-user GPU serving: PagedAttention, continuous batching.

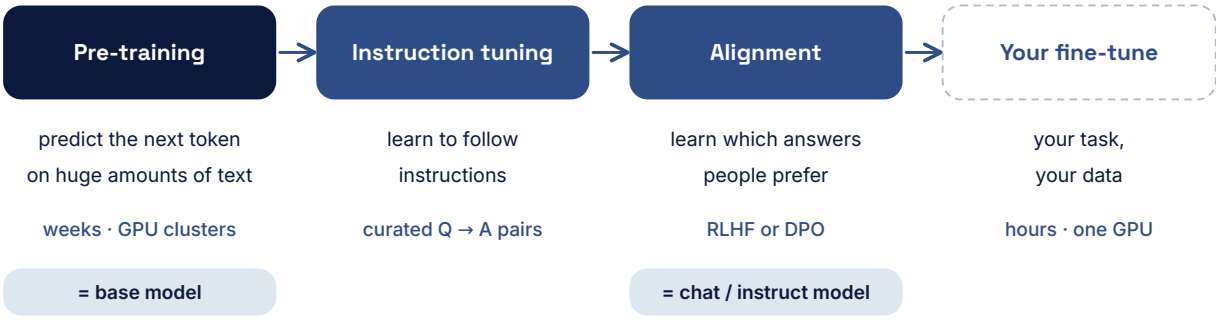
llama.cpp
CPU and edge, GGUF models.

WEB-DEV LENS Like Nginx workers with connection pooling: keep the expensive resource always busy.

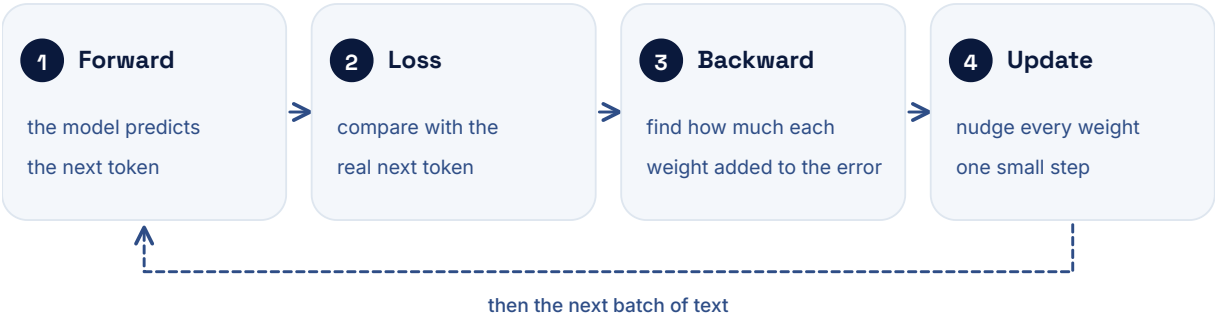
How models are trained

Four stages make a model. One simple loop drives all of them.

HOW A MODEL IS MADE



ONE TRAINING STEP · REPEATED MILLIONS OF TIMES



EXAMPLE

```
"The cat is ___"   model: sitting 0.61   text: sleeping   →   raise p(sleeping)
```

KEY POINTS

- **Base model**: completes text. **Chat model**: follows instructions.
- **Learning rate** = step size. Too big is unstable; too small is slow.
- **Epoch** = one pass over the data.
- **Overfitting** = memorised the training set. Watch the validation loss.

IN PLAIN TERMS

```
loss = -log p(correct token)

weight ← weight - lr × gradient

gradient = each weight's share
           of the error
```

IN PRODUCTION

You rarely pre-train
Start from a pretrained open-weight model.

Validation loss
Stop when it stops improving.

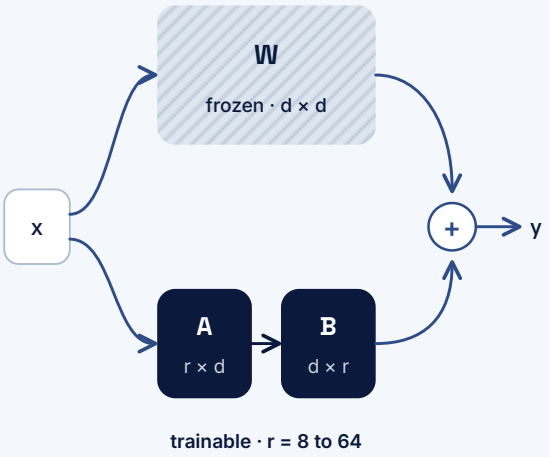
Data quality
For fine-tunes it matters more than data size.

WEB-DEV LENS Like tuning a config by feedback: run, measure the error, adjust, repeat.

Fine-tuning with LoRA

Change how a model behaves by training a small add-on, not the model.

LoRA: train a small add-on



Trains well under 1% of the weights.

The base model itself never changes.

The workflow

1. **Collect examples**
   200–5,000 pairs: prompt → ideal answer
2. **Split the data**
   a training set and a validation set
3. **Train the adapter**
   QLoRA: 4-bit base model, fits one GPU
4. **Evaluate**
   tuned vs base, on your golden set
5. **Serve**
   merge into the base, or load at runtime

Try these first · fine-tune last

Better prompt	Few-shot examples	RAG	Fine-tune
minutes	minutes	days	days to weeks + upkeep

Why it is cheap

$$W' = W + (\alpha / r) \cdot B \cdot A$$

```
d = 4,096 · r = 16
full:  d × d   = 16.8M weights
LoRA:  2 · d · r = 131k (0.8%)
```

Key points

- Fine-tune for **behaviour**: style, format, one narrow task.
- Use RAG for **knowledge**: facts that change.
- A tuned **small** model can replace a large one on a narrow task.

In production

Tools
Hugging Face PEFT and TRL, Unsloth, Axolotl.

Chat template
Train in the same prompt format you serve with.

Regression check
Test general tasks too; tuning can break them.

WEB-DEV LENS Like a plugin on WordPress core: core stays untouched, the plugin changes behaviour.

Choosing a model

Four questions narrow down the options.

WHERE IT RUNS

API model (closed weights)

No servers to run. Top quality.
Data leaves your network. Pay per token.

Open-weight, self-hosted

Data stays with you. Fixed GPU cost.
You run, scale and update it.

SIZE · A ROUGH GUIDE

Small · up to ~8B

routing, extraction,
classification

Medium · ~8–40B

most RAG and
tool calling

Large · 70B+, frontier

hard reasoning,
long agent runs

TYPE

Chat / instruct

text in, text out

Reasoning

thinks first: slower,
better at multi-step

Embedding

text → vector,
for search

Multimodal

also reads images
or audio

READ A MODEL FILE NAME

```
Llama-3.1-8B-Instruct-Q4_K_M.gguf
```

`Llama-3.1`	`8B`	`Instruct`	`Q4_K_M`	`.gguf`
family, version	8 billion weights	tuned for chat	4-bit quantized	llama.cpp / Ollama

KEY POINTS

- Check: **context window**, **licence**, **tool calling**, price or VRAM.
- Choose by **evals on your task**, not by leaderboards.
- Start with the **smallest model** that passes your evals.

WILL IT FIT?

```
VRAM ≈ params × bytes + KV cache

8B at 4-bit ≈ 5 GB + cache
```

IN PRODUCTION

Licence
Open-weight is not always open-source. Read the terms.

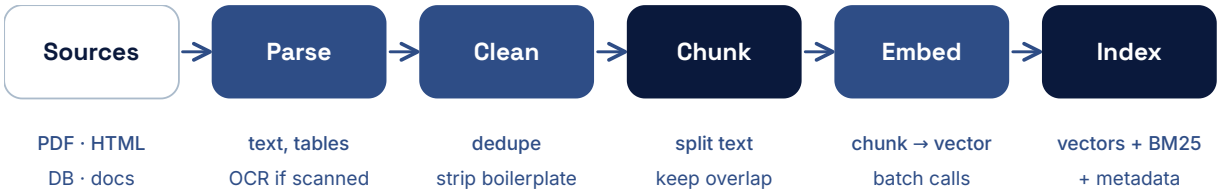
Retirement
API models get retired. Pin versions; re-run evals.

Two-model setup
Small model for easy calls, large for hard ones.

WEB-DEV LENS Like shared hosting versus your own server: convenience against control.

RAG: ingestion

Everything that happens before the first question is asked.



THREE WAYS TO CHUNK

Fixed-size



equal blocks + overlap

Simple and fast.

Can cut mid-sentence.

Recursive

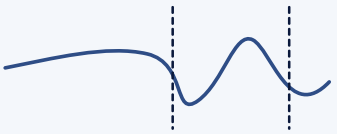


headings → paragraphs → sentences

Respects structure.

A good default.

Semantic



cut where similarity drops

Better boundaries.

Costs extra embeddings.

ONE CHUNK RECORD

```
{ text, vector, source, page, section, acl, updated_at }
```

CHUNK COUNT

```
chunks = ceil(
  (N - overlap) / (size - overlap) )

N=10,000 · size=500 · overlap=50
  → 23 chunks
```

KEY POINTS

- Common size: **256 – 1,024 tokens**, overlap 10–15%.
- Tune on **retrieval recall**, not on feel.
- Metadata powers **filters and citations**.
- **Contextual retrieval**: prepend a short doc summary to each chunk before embedding.
- **Parent-child**: embed small chunks, return their parent section to the LLM.

IN PRODUCTION

Garbage in
Bad parsing means bad chunks means wrong answers.

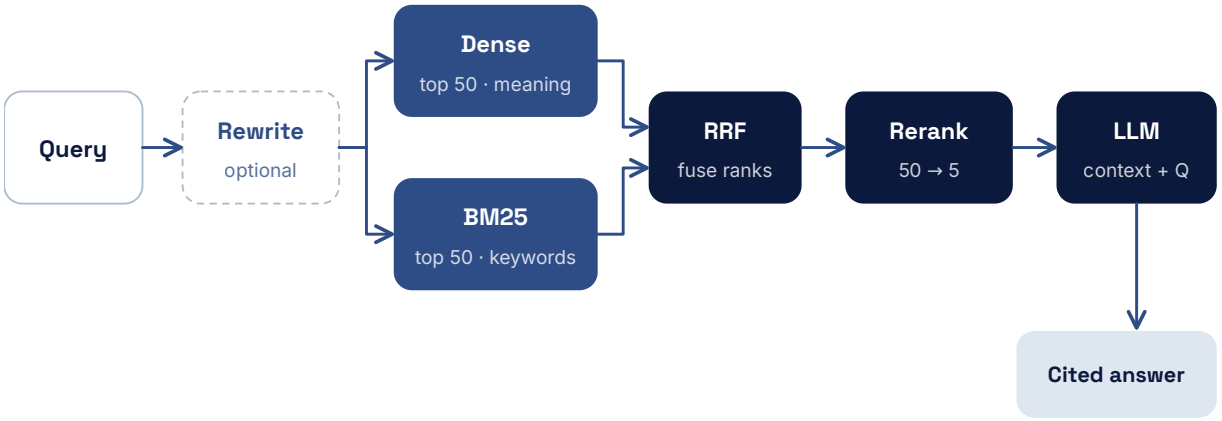
Access control
Store the ACL on each chunk; filter at query time.

Re-ingest
Version documents; update or delete stale vectors.

WEB-DEV LENS Like an ETL job feeding a search index: extract, clean, split, load.

RAG: retrieval

Hybrid search, fused, then re-ranked, then handed to the model.



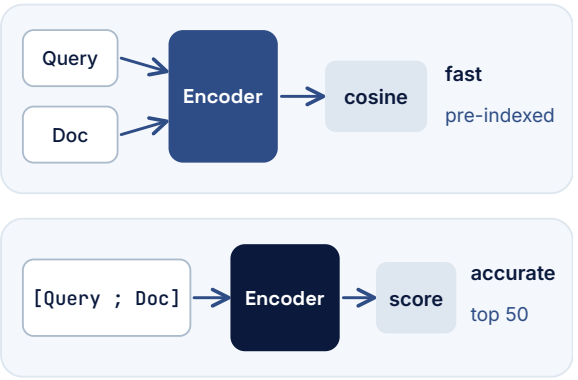
RRF · WORKED EXAMPLE

doc	dense rank	BM25 rank	score
A	1	3	0.0323
B	2	1	0.0325

B's combined rank (2, 1) beats A's (1, 3).

k = 60 damps the gap between ranks 1 and 2.

BI-ENCODER VS CROSS-ENCODER



RECIPROCAL RANK FUSION

$$RRF(d) = \Sigma\ 1\ /\ (k + rank_i(d))$$

k ≈ 60 · sum over each ranked list i

KEY POINTS

- **Retrieve wide** (50), **rerank narrow** (5).
- Dense finds meaning. BM25 finds exact terms: names, SKUs, error codes.

IN PRODUCTION

Filters first
Apply tenant, ACL and date filters inside the search.

Latency
Rerank adds roughly 100–300 ms; measure it on your data.

Cite
Return chunk IDs so answers can show sources.

WEB-DEV LENS Like MySQL FULLTEXT plus a smarter ORDER BY on the top results.

RAG: patterns

Five options. Add complexity only when your evals show a gap.

FIVE OPTIONS · PICK BY THE PROBLEM YOU HAVE

Naive	START HERE Advanced	Corrective	Agentic	GraphRAG
retrieve top-k then generate	rewrite query hybrid search rerank	grade results bad? rewrite or search web	agent decides when and where to retrieve	entity graph + summaries whole-corpus Qs
COST ●○○○○	COST ●●○○○	COST ●●●○○	COST ●●●●○	COST ●●●●○
FAQ, small corpus	Production default	Noisy corpus	Multi-hop questions	Corpus themes

DECISION RULE

- **Start at Advanced.** It covers most business use cases.
- Add **Corrective** when retrieval is unreliable.
- Add **Agentic** when one question needs several lookups.
- Add **GraphRAG** when answers span the whole corpus, not a few chunks.
- **GraphRAG** and **Agentic** solve different problems; they are not rungs on one ladder.

IN PRODUCTION

Every addition brings latency, cost and new failure modes.

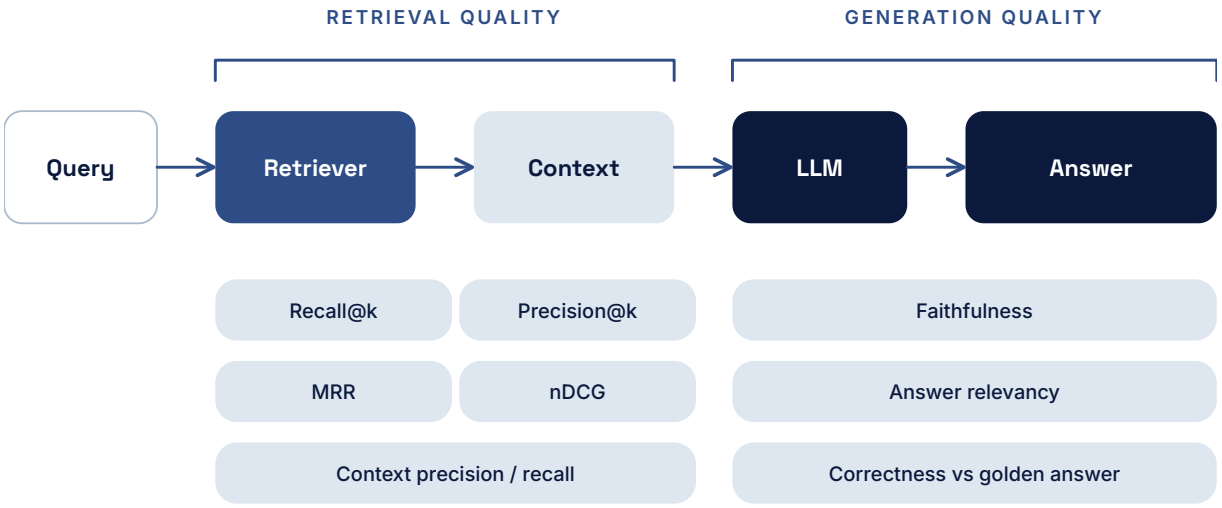
Corrective needs a reliable relevance grader.

GraphRAG has a heavy indexing cost; budget for it.

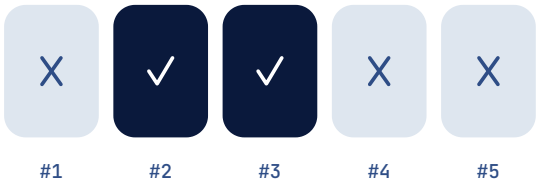
WEB-DEV LENS Like caching layers: add each one only when measurements justify the complexity.

RAG: evaluation

Test the retriever and the generator separately.



WORKED EXAMPLE · TOP-5 RESULTS · 3 RELEVANT DOCS EXIST



0.67	0.40	0.50	0.53
Recall@5	Precision@5	MRR	nDCG@5

RETRIEVAL METRICS

```
hits@k = relevant docs in top k
Recall@k    = hits@k / all relevant
Precision@k = hits@k / k
MRR = mean(1 / first-hit rank)
nDCG@k = DCG@k / IDCG@k
DCG = Σ rel_i / log_2(i + 1)
```

GENERATION METRIC

```
Faithfulness =
  claims supported by context
  ÷ all claims in the answer
```

IN PRODUCTION

Wrong answer, right context
A prompt or model problem.

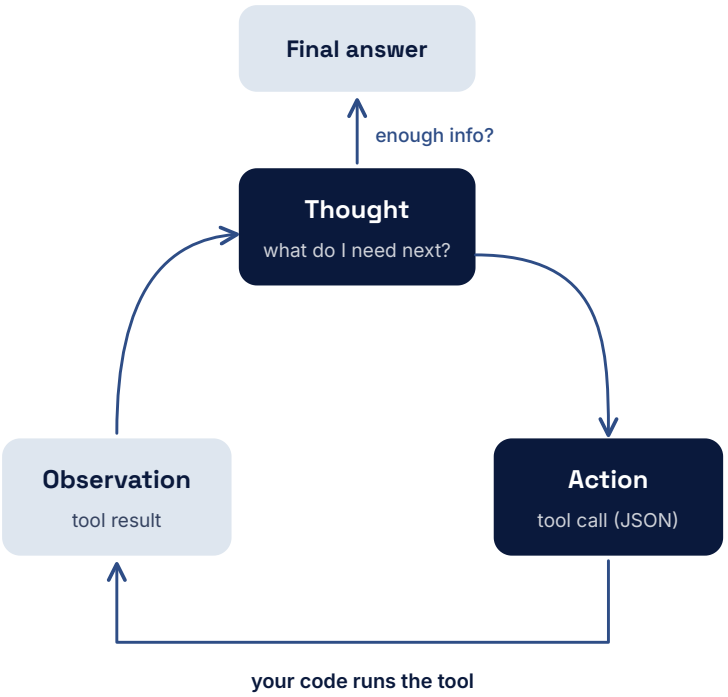
Wrong answer, wrong context
A retrieval problem.

Build a golden set
50–200 real questions with known sources.

WEB-DEV LENS Like testing the SQL query and the template separately.

The ReAct agent loop

Think, act, observe. Repeat until done.



MODEL OUTPUT · TOOL CALL

```
{
  "tool": "search_orders",
  "args": {
    "customer_id": "C-1042",
    "status": "open"
  }
}
```

OBSERVATION · FED BACK

```
[{"id":"A-77",
  "total":120.5}]
```

Model proposes. Your code executes. The result goes back in.

GUARDS EVERY LOOP NEEDS

- **max_steps** and a timeout on every run.
- **Validate arguments** against a schema (Pydantic) before executing.
- **Allow-list** tools; give each the least privilege.
- Make tools **idempotent** so retries are safe.

WHAT BREAKS

- **Infinite loops**: the same call again and again.
- **Invented tools** or malformed arguments.
- **Huge tool output** flooding the context window.

IN PRODUCTION

Native tool calling
Models return structured calls. No text parsing.

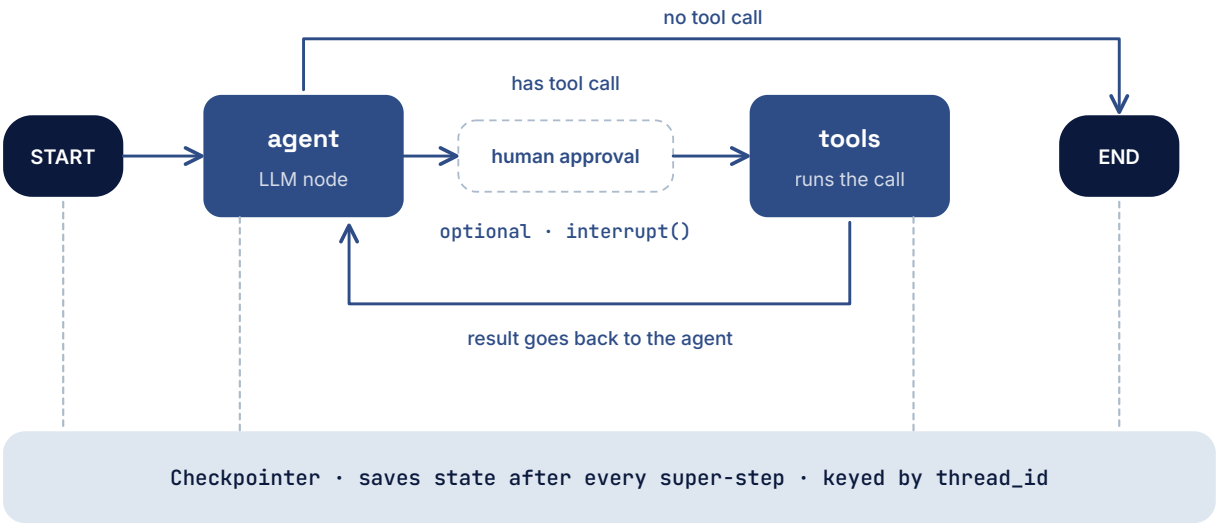
Budget
Cap steps, tokens and wall-clock time per run.

Trace it
Each Thought, Action and Observation is a span.

WEB-DEV LENS Like a controller that calls a service, reads the response, then decides the next call.

LangGraph in one page

An agent is a graph: state flows through nodes along edges.



STATE

```
class State(TypedDict):
    messages: Annotated[
        list, add_messages]
    steps: int
    approved: bool
```

ONE STEP

1. Node reads the state
2. Node returns a partial update
3. A reducer merges it into state
4. Checkpoint is saved
5. An edge picks the next node

MAP TO WHAT YOU KNOW

State
≈ request / session object

Node
≈ controller function

Edge
≈ route

Conditional edge
≈ if / switch

Checkpointer
≈ session store in a database

thread_id
≈ session ID

IN PRODUCTION

create_agent
From langchain.agents. Replaces create_react_agent.

Durable checkpointer
Use a database (e.g. Postgres) in production, not memory.

interrupt() + Command
Pause for approval; resume with Command(resume=...).

Workflows vs agents

Five patterns from Anthropic's guide. Most production work is a workflow.

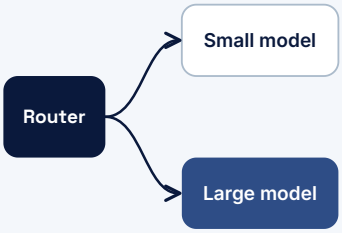
Prompt chaining



a check runs between steps

each output feeds the next

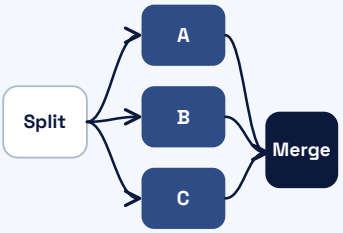
Routing



easy → small · hard → large

classify · cf. Router, p. 17

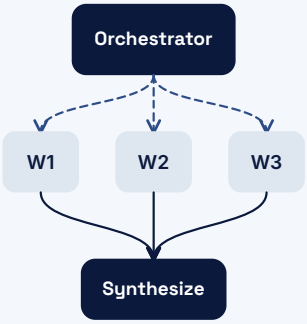
Parallelization



run at once, then merge

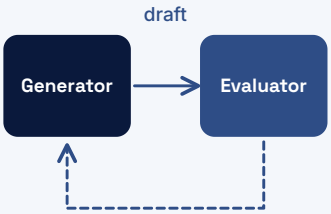
split, run together, merge

Orchestrator-workers



aka supervisor, hierarchical

Evaluator-optimizer



feedback until it passes

aka reflection, self-critique

THE TEST

Code decides the path:

that's a workflow.

The LLM decides the path:

that's an agent.

Start with workflows.

DESIGN RULES

- **Start with the simplest** pattern that meets the requirement.
- Workflows are **predictable and cheaper to test**; agents trade that for flexibility.
- Evaluator-optimizer needs a **reliable critic**. A weak critic loops without end.
- Orchestrator-workers fits when the subtasks **can't be known in advance**.

IN PRODUCTION

Gate each step
Validate output before the next call.

Count calls
Cost ≈ calls × tokens per call.

Cap agents
Set max_steps. A loop without a cap is an open bill.

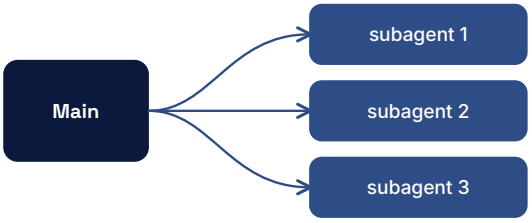
WEB-DEV LENS

Like a shell script (workflow) versus a daemon that picks its own next command (agent).

Subagents, handoffs, skills, router

The four multi-agent building blocks in current LangChain docs.

Subagents



Main agent calls subagents as tools.

Several can run in one turn.

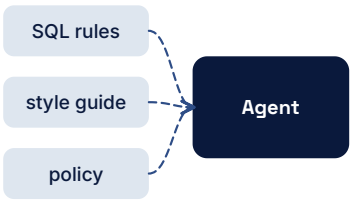
Handoffs



Current agent transfers control via state.

The next agent talks to the user.

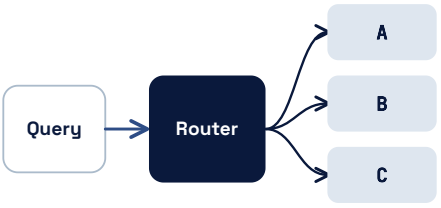
Skills



One agent loads specialist prompts on demand.

Context grows with each skill loaded.

Router



A classifier sends work to specialist agents.

Like Routing (p. 16), but targets are agents.

CHOOSING A PATTERN

- **Parallel domains**: subagents or router. **Repeat requests**: handoffs and skills keep state.
- A **supervisor** is a subagent setup where the main agent keeps conversation context. A **router** is one classification step.
- **Plan-and-execute**: a planner writes steps, an executor runs them, and the plan is revised after failures.
- Pattern names drift between frameworks. Match on behaviour, not on the label.

IN PRODUCTION

Context isolation
Subagents see only the context they need.

Repeat calls
Stateful patterns reuse context on repeat requests.

Trace control
Log which agent holds control at each step.

WEB-DEV LENS Like splitting one service into controllers, each owning one responsibility.

How agents act and plan

Four ways an agent turns a decision into action.

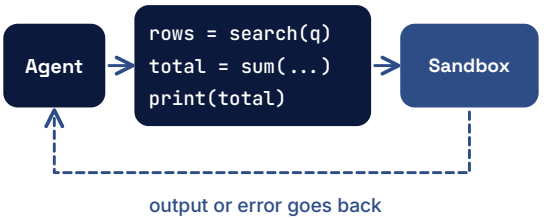
Tool calling (JSON)



Model picks one tool and its arguments.

Safe, simple, easy to validate. The default.

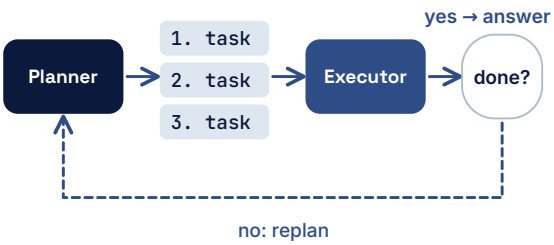
CodeAct



Model writes code that calls several tools at once.

Fewer steps, loops and maths. Must run sandboxed.

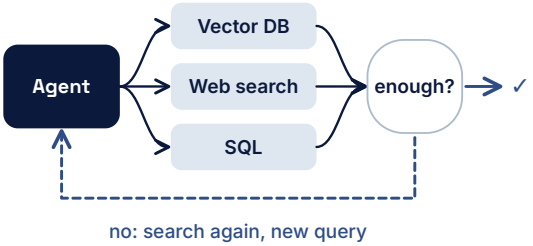
Plan-and-execute



Plan first, then run tasks one by one.

Long tasks with clear steps. Cap replans.

Agentic RAG



Agent chooses where and how often to retrieve.

Multi-source, multi-hop questions.

WHICH ONE WHEN

- Start with **JSON tool calls**.
- Use **CodeAct** when a task needs loops, maths or many tool calls in a row.
- Use **planning** when the task is long and its steps are clear upfront.
- Use **agentic RAG** when one search isn't enough.

SAME IDEA, OTHER NAMES

- **Corrective RAG** is often drawn as "self-reflection". See RAG patterns.
- **Reflection** = Evaluator-optimizer. See Workflows vs agents.
- **Tool use via MCP**. See Memory & MCP.

IN PRODUCTION

Sandbox CodeAct
Container, no network by default, CPU and time limits.

Cap the plan
Limit replans and total steps.

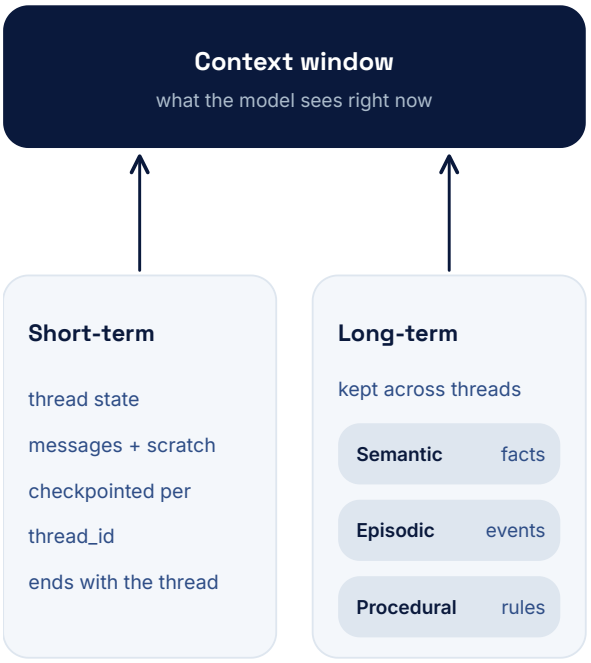
Stop retrieval
Max searches per question, then answer or say "not found".

WEB-DEV LENS JSON call ≈ one API request. CodeAct ≈ the agent writes a small script that makes many requests.

Memory & MCP

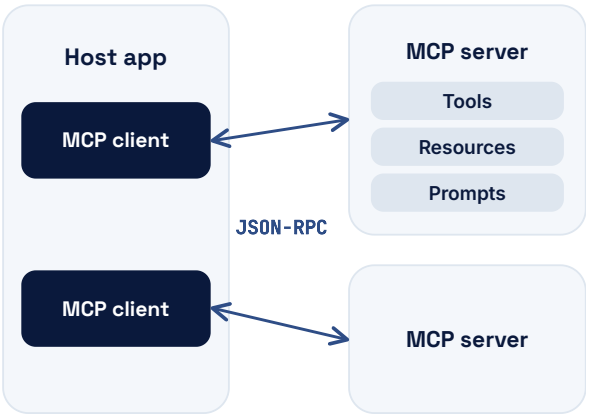
What an agent remembers, and how it reaches the outside world.

MEMORY



Write only what is worth keeping. Summarise old turns.

MCP · MODEL CONTEXT PROTOCOL



Tools	actions the model can call
Resources	data it can read
Prompts	reusable templates

Transports: stdio (local) · Streamable HTTP (remote)

MEMORY

- Short-term memory comes with a **checkpointer**.
- Long-term memory needs **your own** write and read policy.

MCP

- Write a tool once; **any MCP host** can use it.
- Spec 2026-07-28: **stateless**. No handshake, no session ID.
- Treat server output as **untrusted input**.

IN PRODUCTION

Context budget
Memory shares the window. Retrieve; don't dump.

Remote MCP
Use OAuth, least privilege, and log every tool call.

No secrets
Keep keys and personal data out of long-term memory.

WEB-DEV LENS Memory ≈ session vs database. MCP ≈ an API contract for tools, like OpenAPI.

Context engineering

Decide what the model sees at each step. Nothing else reaches it.

THE CONTEXT WINDOW



FOUR MOVES

Write

Save notes outside the window: scratchpad, files, memory.

Select

Pull in only what this step needs: RAG, tool search.

Compress

Summarise or trim old turns and long tool output.

Isolate

Give a subagent its own clean window.

WHY IT MATTERS

- More context is **not** better. Long prompts dilute attention.
- Put **stable content first** so prompt caching can reuse it.
- **Tool output** bloats context fastest. Truncate or summarise it.

HOW CONTEXT FAILS

- **Poisoning**: a wrong fact gets in and is reused.
- **Distraction**: too much history; the model repeats old actions.
- **Clash**: two sources disagree.

IN PRODUCTION

Measure it
Log tokens per section of the prompt.

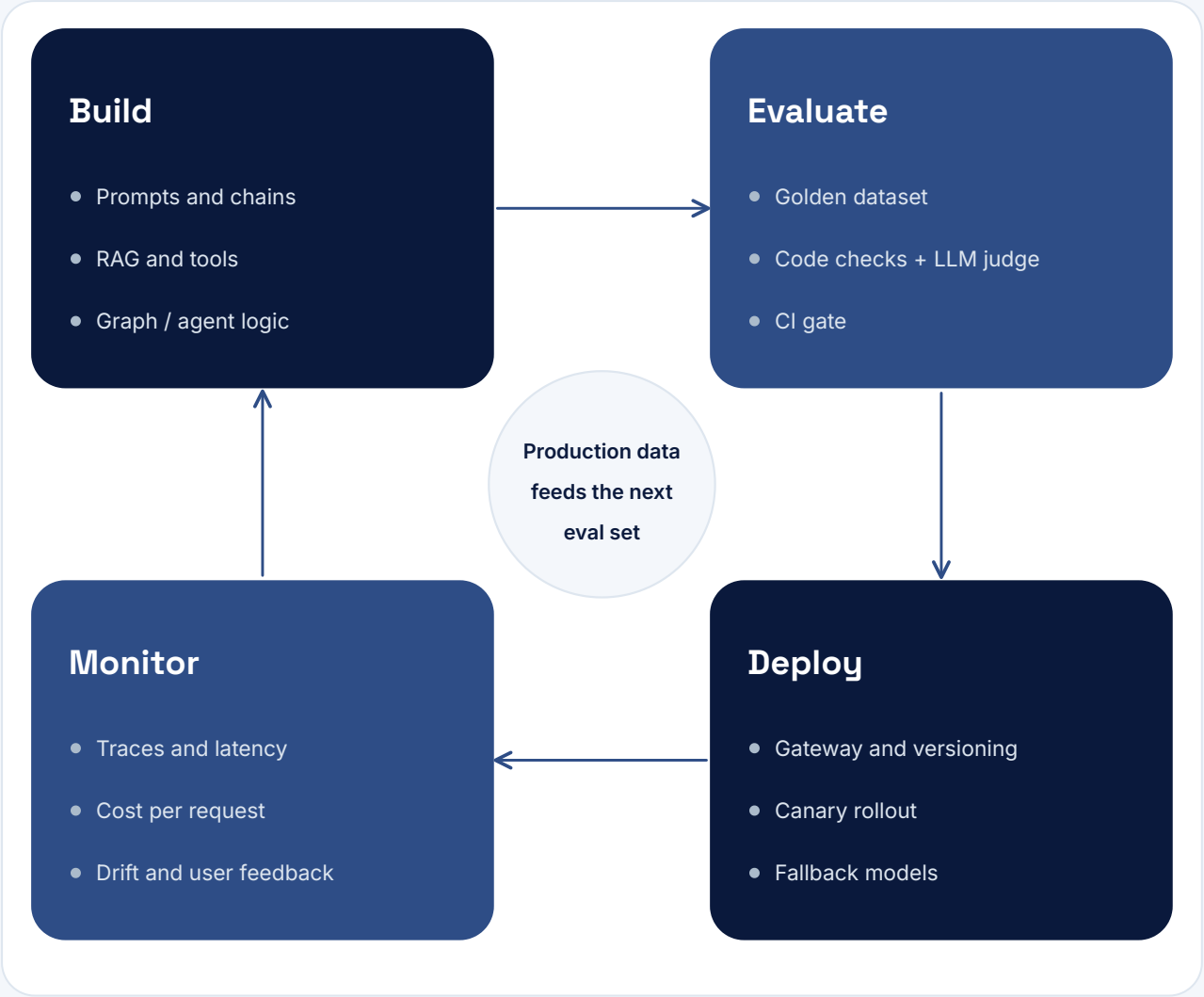
Cache the prefix
Stable parts first, changing parts last.

Budget tools
A few clear tools beat many overlapping ones.

WEB-DEV LENS Like choosing what to load per request: the whole database never fits in RAM.

The LLMOps lifecycle

Build, evaluate, deploy, monitor. Then loop.



RELIABILITY CHECKLIST

- **Pin** model and prompt versions.
- Keep **temperature low** for deterministic steps; some reasoning models reject it.
- **Schema-validate** every output.
- **Retry, then fall back** to another model.

RULES OF THUMB

- **No eval, no deploy.**
- Every incident becomes a test case.
- Version prompts like code.

IN PRODUCTION

Deterministic where possible
Use code for what code can do.

Gateway
One place for auth, limits, logging and fallbacks.

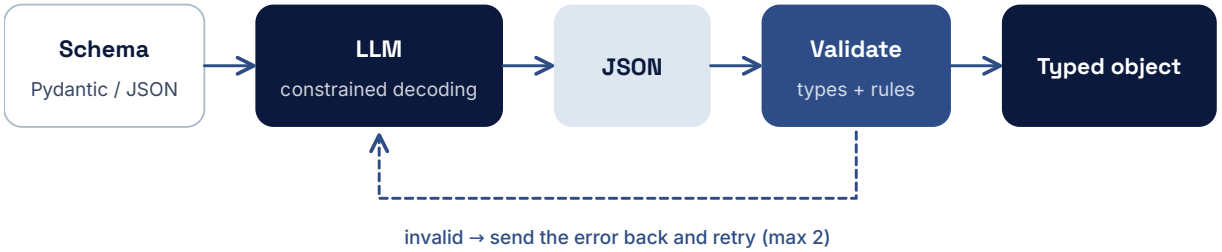
Feedback loop
Failed traces become new golden examples.

WEB-DEV LENS Like CI/CD plus APM: tests gate the release, monitoring feeds the next test.

Structured output

Get data your code can trust, not text you have to parse.

FROM SCHEMA TO TYPED OBJECT



THREE LEVELS

Prompt only

"Reply in JSON."

Can break or add prose.

RELIABILITY

●●○○○

JSON mode

Always valid JSON.

Shape is not guaranteed.

RELIABILITY

●●●○○

Schema-constrained

Valid JSON that always matches your schema.

RELIABILITY

●●●●●

SCHEMA

```
class Invoice(BaseModel):
    vendor: str
    total: float
    status: Literal["paid", "due"]
```

OUTPUT

```
{ "vendor": "Acme",
  "total": 120.5,
  "status": "due" }
```

KEY POINTS

- **Constrained decoding** blocks tokens that would break the schema, so the syntax cannot fail.
- Right shape is not right values. Still check **ranges and business rules**.
- Keep schemas **small and flat**. Describe each field.

IN PRODUCTION

Tool calls
Arguments are structured output too. Validate them.

Retries
Feed the validation error back; cap at two.

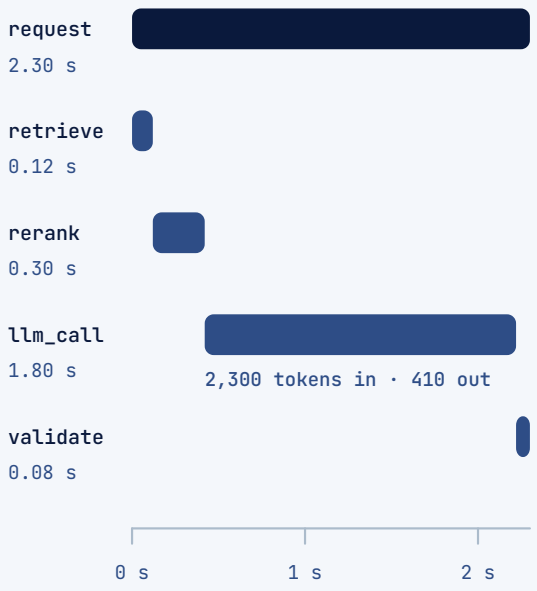
Enums
Use fixed choices for categories, not free text.

WEB-DEV LENS Like server-side form validation: never trust the shape of incoming data.

Tracing & evals

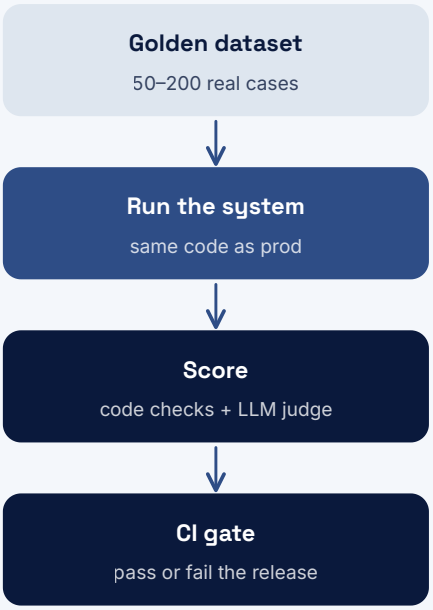
See what happened. Prove it still works.

ONE TRACE · 2.3 s



Each bar is a span; the tree is the trace.

OFFLINE EVAL → CI GATE



TOOLS AND HABITS

- **LangSmith**, **Arize Phoenix**, **Helicone**: traces, datasets, evals.
- Log tokens, latency and cost on **every span**.
- Sample production traces into **review queues**.
- Emit **OpenTelemetry GenAI** spans so any backend can read them.

LLM-AS-JUDGE CAUTIONS

- Calibrate the judge against **human labels**.
- Watch for **position** and **length** bias.
- **Pin** the judge model and its prompt.

IN PRODUCTION

Online evals
Score a sample of live traffic, not just the test set.

Regression tests
Every bug becomes a permanent dataset row.

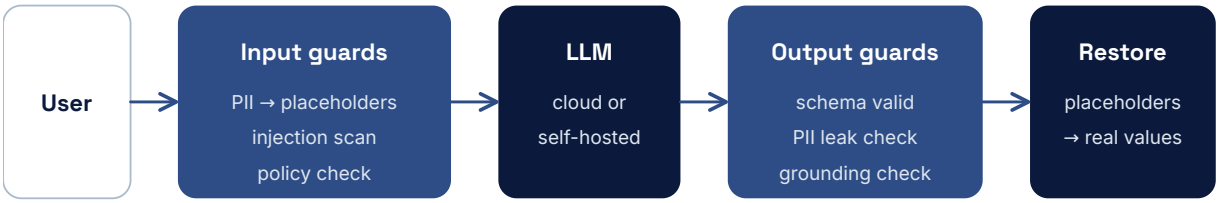
Cost per trace
Attach tokens and price to each run.

WEB-DEV LENS Like logs plus unit tests: traces show what ran, evals say whether it was right.

Guardrails & privacy

Check what goes in. Check what comes out.

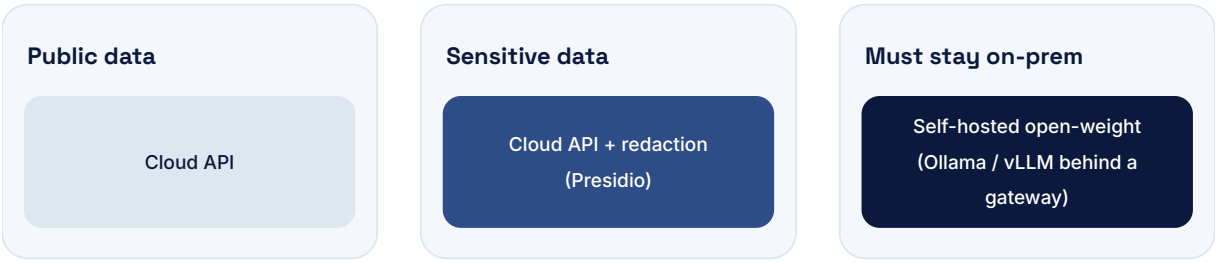
ONE REQUEST, END TO END



THE SAME REQUEST, THREE VIEWS



WHERE SHOULD THE MODEL RUN?



INJECTION & DETECTION

- Treat **retrieved text, web pages and tool output** as untrusted.
- **Least privilege** for every tool.
- **Human approval** before irreversible actions: payments, deletes, emails.
- Lethal trifecta: private data + untrusted content + an outbound channel. Remove one.
- Test PII detection on your own names and languages; add custom recognizers.

IN PRODUCTION

Keep the map local
The placeholder-to-value table never leaves your side.

Never log raw PII
Redact before traces and logs too.

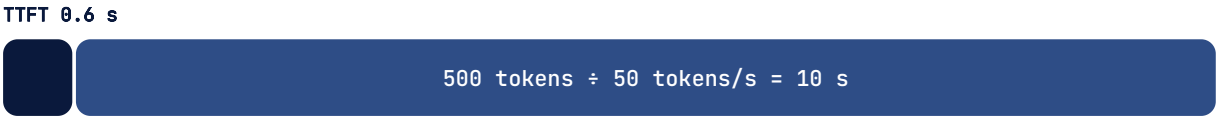
Test the guards
Run adversarial prompts in CI.

WEB-DEV LENS Like input sanitising and output escaping around a database call.

Cost & latency

Two formulas explain almost every bill and every wait.

ONE REQUEST · EXAMPLE



Total ≈ 10.6 s. Streaming shows the first words after 0.6 s, so it feels fast.

COST · EXAMPLE PRICES: $3 PER 1M INPUT, $15 PER 1M OUTPUT TOKENS

Input	+	Output	=	Per call	×	100k calls
$0.0060		**$0.0075**		**$0.0135**		**$1,350**
2,000 tokens × $3 / 1M		500 tokens × $15 / 1M		input + output		per month at this rate

SIX LEVERS

Prompt caching
Repeated prefixes cost less and start faster.

Shorter context
Retrieve less, rerank, trim history.

Model routing
Easy tasks go to a small model.

Semantic cache
Reuse answers to near-duplicate questions.

Streaming
Lower perceived latency; same total time.

Batch jobs
Offline work can use cheaper batch APIs.

FORMULAS

$$\text{cost} = \text{in} \cdot p_{in} + (\text{out}+\text{think}) \cdot p_{out}$$

$$\text{latency} \approx \text{TTFT} + \text{out} / (\text{tokens/s})$$

KEY POINTS

- Output tokens usually cost **more** than input tokens.
- Long prompts raise TTFT; long answers raise **total time**.
- **Reasoning (thinking) tokens** bill as output. Set a thinking budget.

IN PRODUCTION

Budgets
Set max_tokens and per-user limits.

Fallbacks
Cheaper model when the primary fails or is slow.

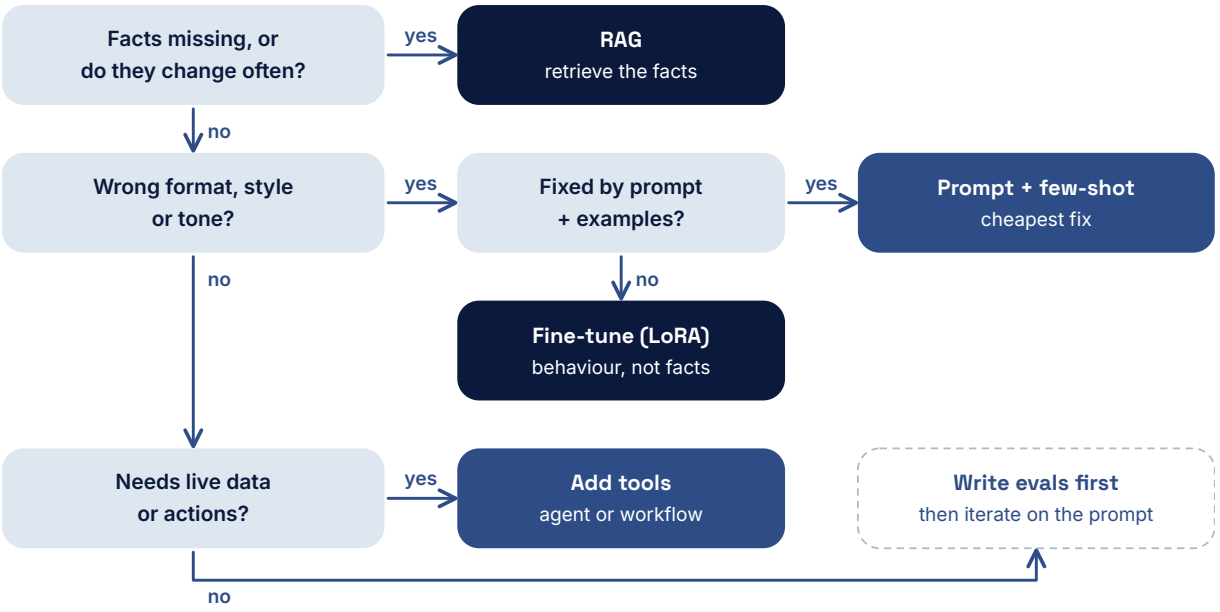
Watch p95
Averages hide the slow requests users remember.

WEB-DEV LENS Like database tuning: cache what repeats, fetch less, watch the slow tail.

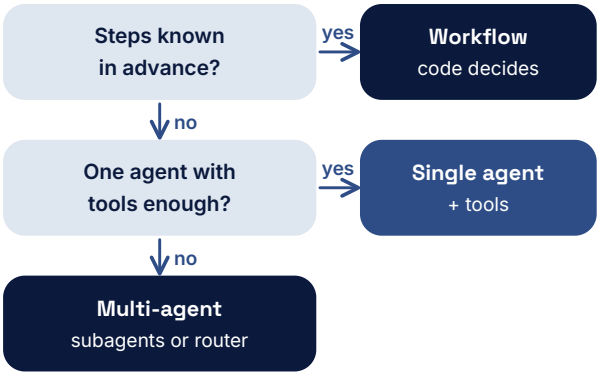
Decision trees

Three yes/no questions that settle most design arguments.

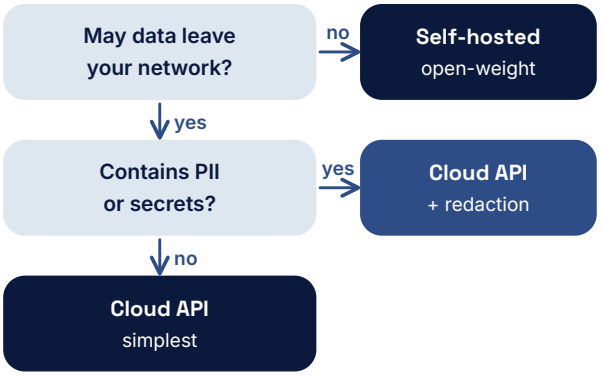
MY SYSTEM GIVES WEAK ANSWERS. WHAT DO I ADD?



WORKFLOW OR AGENT?



WHERE SHOULD IT RUN?



HOW TO USE

- Start at the **top question** and stop at the first **yes**.
- Each tree ends in the **simplest option** that solves the problem.
- Unsure at a question? Build a small eval set and **measure**.

IN PRODUCTION

RAG vs fine-tune
Facts → RAG. Behaviour → fine-tune. Often both.

Workflow vs agent
Known steps → workflow. Unknown → agent.

Revisit
Re-run the tree when evals or data change.

WEB-DEV LENS Like choosing cache vs index vs new table: the question decides, not the trend.

Numbers to know

Typical values, so you can tell normal from broken at a glance.

MODELS & MEMORY

1B params at FP16	≈ 2 GB
7–8B at 4-bit	≈ 4–5 GB
70B at 4-bit	≈ 40 GB
1 token	≈ ¾ word, 4 chars
Context window	128k – 1M tokens

RAG

Embedding size	384 – 4,096 dims
Chunk size	256 – 1,024 tokens
Chunk overlap	10 – 15%
Retrieve → rerank	50 → 5
RRF constant k	60

LATENCY

Good TTFT for chat	< 1 s
Decode speed	30 – 150 tok/s
Vector search	≈ 10 – 50 ms
Reranker	≈ 100 – 300 ms
Embed one query	≈ 10 – 100 ms

EVALS, TUNING, AGENTS

Golden set	50 – 200 cases
Fine-tune data	200 – 5,000 pairs
LoRA rank r	8 – 64
Agent max_steps	10 – 25
Retries on bad output	max 2

READ THESE AS RANGES

- These are **rules of thumb** from common practice in 2026, not guarantees.
- Use them to spot outliers: a 5 s TTFT or a 4,000-token chunk is worth a look.
- Measure **your** system and write your own numbers in the margin.

IN PRODUCTION

Memory
Add KV cache and overhead on top of weights.

Latency
Watch p95, not the average.

Cost
Recheck prices; they drop often.

WEB-DEV LENS Like knowing a healthy page load or query time before you start tuning.

Anti-patterns

Common mistakes, and what to do instead.

DON'T	DO INSTEAD
✗ Start with many agents	✓ One agent + tools; split when evals fail
✗ Paste whole documents in	✓ Retrieve and rerank only what is needed
✗ Ship without evals	✓ Golden set + CI gate before launch
✗ Let the agent loop freely	✓ max_steps, timeout, token budget
✗ Parse free-text replies	✓ Schema-constrained output + validation
✗ Fine-tune to add facts	✓ RAG for facts; tune for behaviour
✗ Trust tool and web output	✓ Treat it as untrusted; least privilege
✗ Log raw prompts with PII	✓ Redact before logs and traces
✗ Pick a model by leaderboard	✓ Pick by your evals; smallest that passes

PATTERN BEHIND THE MISTAKES

- Most of these come from **skipping measurement**: no evals, no traces, no limits.
- When something breaks in production, check this list first.

IN PRODUCTION

Review
Use this list as a pre-launch checklist.

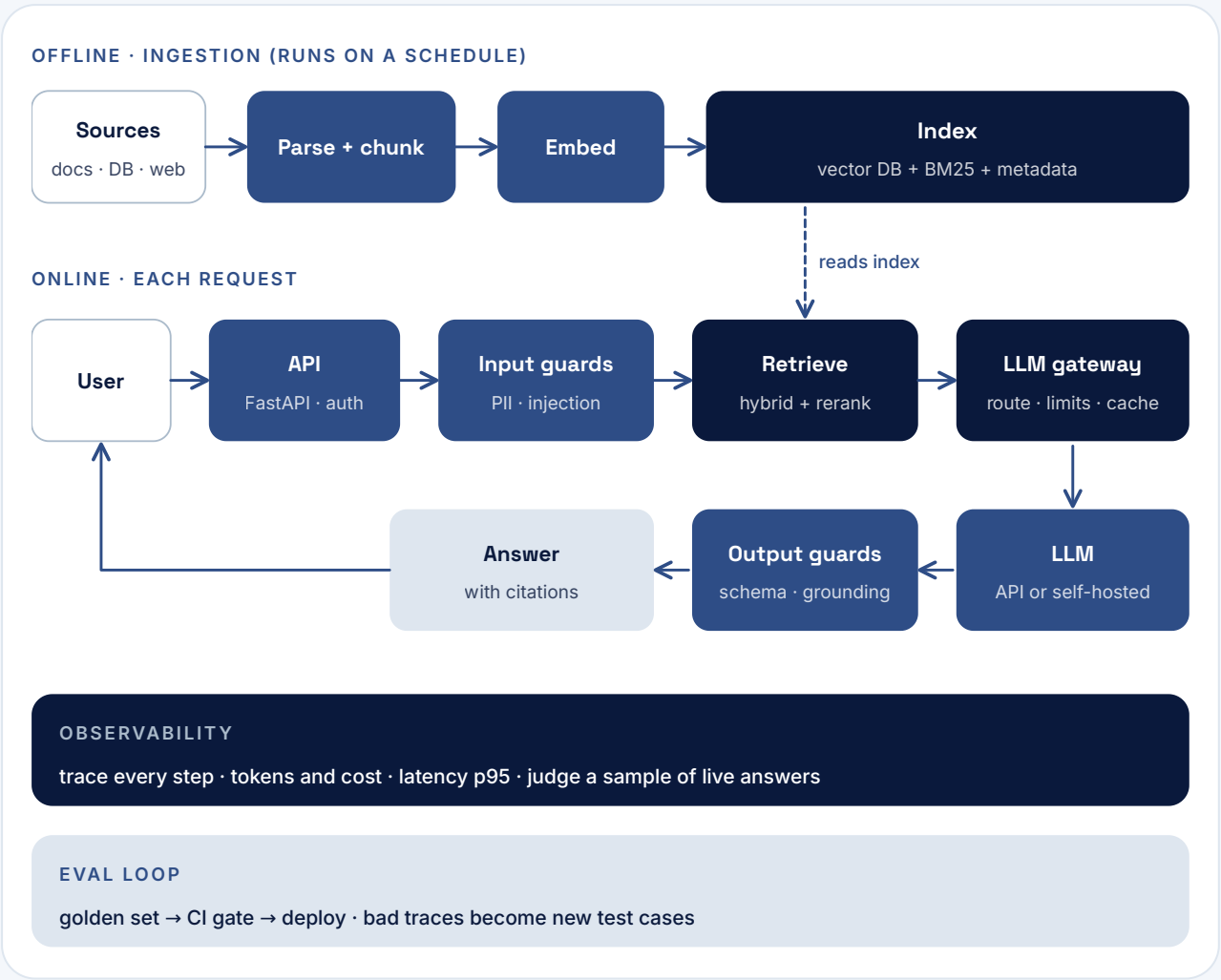
Incidents
Each incident adds a row here and a test case.

Clients
Most rescue jobs start with one of these.

WEB-DEV LENS Like N+1 queries or no input validation: easy to ship, painful to fix later.

Reference: enterprise RAG chatbot

Every piece from Parts 2 and 4, wired into one system.



DESIGN CHOICES

- Ingestion and serving **scale separately**.
- Every answer cites **chunk IDs**.
- Guards run on **both sides** of the model.

EXAMPLE STACK

- FastAPI · Qdrant or pgvector
- Cross-encoder reranker
- LiteLLM-style gateway
- LangSmith or Phoenix · Presidio

IN PRODUCTION

Access control
Filter by the user's ACL inside retrieval.

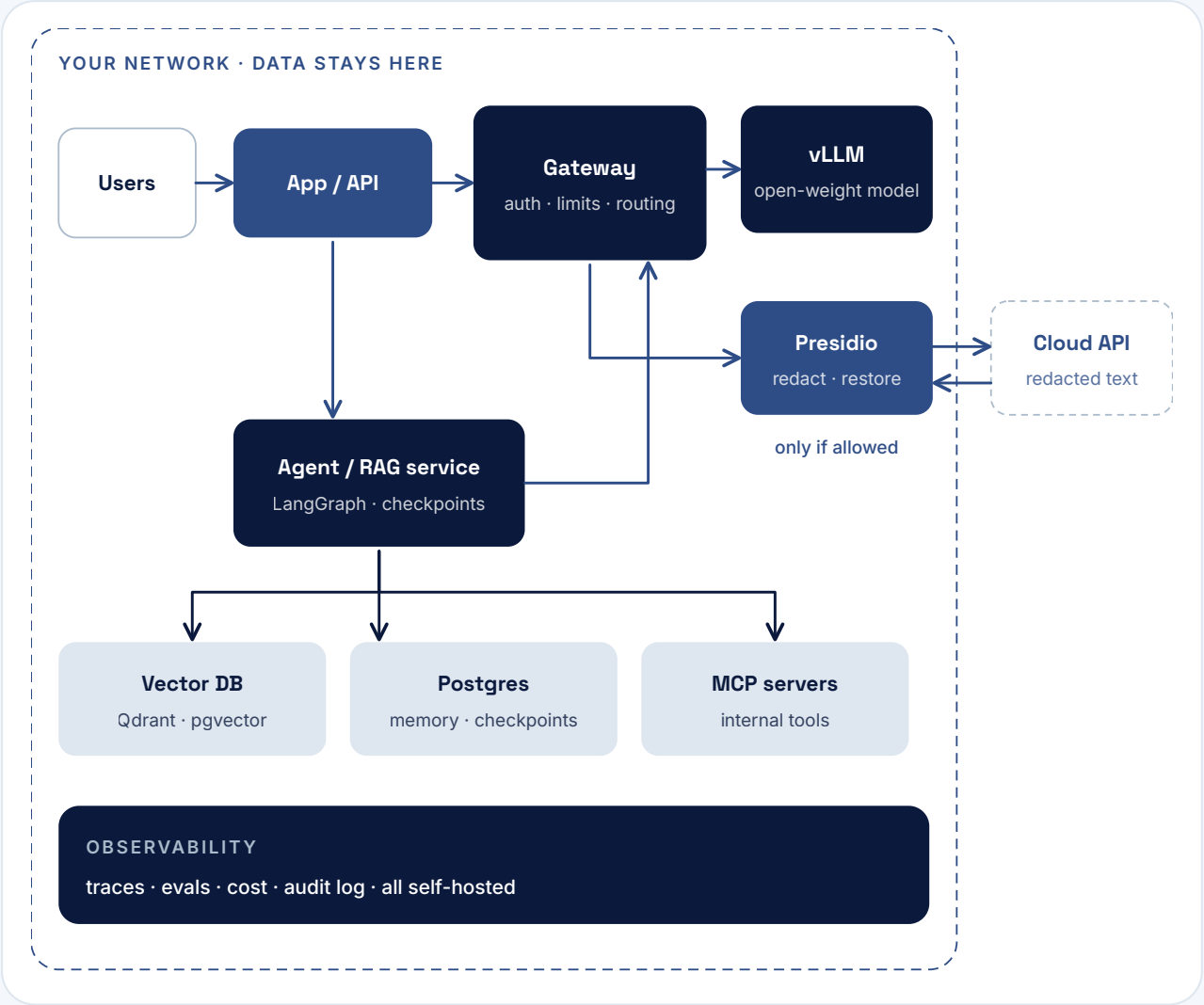
Freshness
Re-ingest changed docs; delete stale vectors.

Fallback
Gateway switches models on errors or slowness.

WEB-DEV LENS Like a classic web app: cron jobs build the index, the request path stays fast.

Reference: private AI stack

A self-hosted setup where sensitive data never leaves the network.



DESIGN CHOICES

- Everything inside the line runs on **your servers**.
- The **gateway** is the one control point for every model call.
- Cloud calls are **optional** and see only redacted text.

EXAMPLE STACK

- GPU box: vLLM or Ollama
- Gateway with auth and quotas
- Phoenix or Langfuse, self-hosted
- Backups for Postgres and vectors

IN PRODUCTION

Sizing
Size GPUs for peak concurrent users, not total users.

Updates
Pin model versions; re-run evals before upgrading.

Audit
Log who asked what, with PII redacted.

WEB-DEV LENS Like an on-prem LAMP stack behind a reverse proxy: one door in, everything else private.

REFERENCE

Formula cheat sheet

Twenty formulas that cover most of this book.

DOT PRODUCT

$a \cdot b = \Sigma\, a_i\, b_i$

similarity before normalising

COSINE SIMILARITY

$\cos\theta = a \cdot b \,/\, (|a|\,|b|)$

−1 to 1; 1 = same direction

SOFTMAX

$\text{softmax}(z)_i = \exp(z_i) \,/\, \Sigma_j \exp(z_j)$

logits → probabilities

TEMPERATURE

$p_i = \text{softmax}(z_i \,/\, T)$

T < 1 sharper, T > 1 flatter

ATTENTION

$\text{softmax}(QK^T \,/\, \sqrt{d_k} + M)\, V$

M = causal mask

RMSNORM

$x \,/\, \sqrt{(\text{mean}(x^2) + \varepsilon)} \cdot g$

stabilises activations

CROSS-ENTROPY

$L = -(1/N)\, \Sigma\, \log\, p(x_t \mid x_{<t})$

next-token training loss

PERPLEXITY

$PPL = \exp(L)$

lower is better

GRADIENT DESCENT

$\theta \leftarrow \theta - \eta\, \nabla L$

η = learning rate

LORA

$W' = W + (\alpha \,/\, r) \cdot B \cdot A$

only A and B train; $r \ll d$

RRF

$RRF(d) = \Sigma\, 1 \,/\, (k + \text{rank}_i(d))$

k ≈ 60

RECALL@K

hits@k / all relevant

Precision@k = hits@k / k

MRR

mean(1 / rank of first hit)

averaged over queries

NDCG@K

DCG@k / IDCG@k

$DCG = \Sigma\, rel_i \,/\, \log_2(i + 1)$

KV CACHE

$2 \cdot L \cdot H_{kv} \cdot d_{head} \cdot n \cdot \text{bytes}$

per sequence

WEIGHTS MEMORY

params × bytes per param

FP16 = 2 · INT8 = 1 · 4-bit ≈ 0.5

COST PER CALL

$\text{in} \cdot p_{in} + (\text{out} + \text{think}) \cdot p_{out}$

thinking tokens bill as output

LATENCY

TTFT + out / (tokens/s)

plus network and queue

CHUNK COUNT

ceil((N - ov) / (size - ov))

N tokens, ov = overlap

FAITHFULNESS

supported claims / all claims

answer checked against context

SYMBOLS

- **n** number of tokens
- **d** vector size
- **d_k, d_{head}** size per attention head
- **L** layers or loss
- **H_{kv}** KV heads
- **T** temperature
- **η** learning rate
- **θ** model weights
- **∇L** gradient of loss
- **k** results kept, or RRF constant
- **r, α** LoRA rank and scale
- **p_{in}, p_{out}** price per token

REFERENCE

Glossary A–Z

236 terms. → points to related terms and the page with the diagram.

A

A2A (Agent2Agent) AGENT
Open protocol for agents from different vendors to find and talk to each other.
→ Multi-agent, Subagent, Supervisor · p. 17

ACL (access control list) SEC
Who may see what. In RAG it is stored on each chunk and filtered at query time.
→ Qdrant, pgvector, Chunking · p. 10

Activation function DL
Non-linear function (ReLU, GELU, SwiGLU) inside a network that lets it learn complex patterns.
→ ReLU, Sigmoid, SwiGLU

Adam optimizer TRAIN
Adaptive optimizer combining momentum with per-weight learning rates.
→ Pre-training, Instruction tuning, Alignment · p. 7

Adapter TRAIN
Small trainable module added to a frozen model for cheap fine-tuning.
→ Unsloth, Chat template, Distillation · p. 8

Agent AGENT
An LLM in a loop that chooses actions, calls tools and reads the results to reach a goal.
→ ReAct, Tool calling, Function calling · p. 14

Agent Skills AGENT
Packaged instructions and resources an agent loads on demand.
→ A2A, Multi-agent, Subagent · p. 17

Agentic RAG RAG
RAG where an agent decides when, what and how often to retrieve.
→ GraphRAG, Knowledge graph, Grounding · p. 12

Alignment TRAIN
Training a model to follow human intent and safety norms, e.g. with RLHF or DPO.
→ RLHF, DPO, Base model · p. 7

ANN (approximate nearest neighbour) RAG
Vector search that trades a little recall for much more speed.
→ ef_search, Vector database, Matryoshka embedding · p. 5

Attention LLM
Lets each token weigh every other token. softmax(QK^T / sqrt(d_k)) V.
→ Self-attention, Multi-head attention, GQA · p. 4

Autoregressive LLM
Generating one token at a time, each conditioned on all earlier tokens.
→ Decoding, Log-probability, Token · p. 3

B

Backpropagation DL
Computes the loss gradient for every weight by applying the chain rule backwards.
→ Learning rate, Epoch, Batch size · p. 7

Base model LLM
A model after pre-training only. It completes text but is not tuned to follow instructions.
→ Loss function, Cross-entropy, Perplexity · p. 7

Batch normalization DL
Normalises activations across a batch to stabilise training.
→ Residual connection, Layer normalization, RMSNorm

Batch size TRAIN
Number of examples processed before one weight update.
→ Validation loss, Overfitting, Underfitting · p. 7

Beam search LLM
Keeps several candidate sequences at each step and returns the best overall.
→ Autoregressive, Decoding, Log-probability · p. 3

Benchmark EVAL
A fixed test set used to compare models, e.g. MMLU.
→ Trace, Span, Observability · p. 23

Bi-encoder RAG
Encodes query and document separately into vectors. Fast and indexable.
→ Cross-encoder, Query rewriting, HyDE · p. 11

Bias–variance tradeoff ML
Simple models underfit (high bias); complex ones overfit (high variance).
→ Regularization, Hyperparameter, Feature engineering

BM25 RAG
Keyword ranking based on term frequency, term rarity and document length.
→ Hybrid search, RRF, Re-ranking · p. 11

BPE (byte-pair encoding) LLM
Tokenizer that repeatedly merges the most frequent symbol pairs into subwords.
→ Vocabulary, Logits, Softmax · p. 3

C

Canary release OPS
Sending a small share of traffic to a new version before full rollout.
→ Fallback model, Gateway, LiteLLM · p. 21

Glossary A–Z

Continued.

Chain-of-thought LLM
Having a model reason step by step before it answers.
→ Agent, ReAct, Tool calling · p. 14

Chat template LLM
The exact format (roles and special tokens) a model expects for a conversation.
→ Distillation, Fine-tuning, LoRA · p. 8

Checkpoint AGENT
Saved snapshot of agent state after a step. Enables resume, replay and approvals.
→ Super-step, Reducer, thread_id · p. 15

Chunk overlap RAG
Tokens repeated between neighbouring chunks so ideas are not cut in half.
→ Recursive chunking, Semantic chunking, Parent-child chunking · p. 10

Chunking RAG
Splitting documents into pieces that are embedded and retrieved individually.
→ Chunk overlap, Recursive chunking, Semantic chunking · p. 10

Classification ML
Predicting a category label for an input.
→ Linear regression, Logistic regression, Supervised learning

Clustering ML
Grouping unlabelled data by similarity.
→ k-means, PCA, k-NN

CNN (convolutional neural network) DL
Shares filters over local regions; strong for images.
→ RNN and LSTM, KL divergence

CodeAct AGENT
The agent writes and runs code (usually Python) as its action, instead of a single JSON tool call. Needs a sandbox.
→ Sandbox, Plan-and-execute · p. 18

Compaction AGENT
Summarising old conversation turns to free context space.
→ System prompt, Prompt caching, Few-shot · p. 20

Confusion matrix EVAL
Table of true versus predicted classes: TP, FP, FN, TN.
→ ROC-AUC, Precision, Recall

Constrained decoding LLM
Blocking tokens that would break a grammar or schema at each step, so output always parses.
→ Structured output, JSON mode · p. 22

Context engineering AGENT
Choosing what enters the context window: instructions, retrieved data, history, tool output.
→ Context rot, Compaction, System prompt · p. 20

Context precision EVAL
Whether relevant chunks are ranked near the top of the retrieved context.
→ Context recall, Hallucination, RAGAS · p. 13

Context recall EVAL
How much of the information needed for the answer was actually retrieved.
→ Hallucination, RAGAS, Golden dataset · p. 13

Context rot AGENT
Quality drop as the context fills with long or irrelevant content.
→ Compaction, System prompt, Prompt caching · p. 20

Context window LLM
Maximum tokens (prompt plus output) a model can handle at once.
→ Attention, Self-attention, Multi-head attention · p. 4

Contextual retrieval RAG
Prepends a short document summary to each chunk before embedding it.
→ Metadata filtering, ACL, Qdrant · p. 10

Continuous batching INFER
Adding new requests to a running batch at every decoding step.
→ PagedAttention, Quantization, vLLM · p. 6

Cosine similarity MATH
cos θ = a · b / (norm a × norm b). Measures the angle between two vectors.
→ HNSW, ANN, ef_search · p. 5

Cross-encoder RAG
Reads query and document together and outputs a relevance score. Accurate, slower.
→ Query rewriting, HyDE, MMR · p. 11

Cross-entropy MATH
Loss that penalises low probability on the correct token: −log p(correct).
→ Perplexity, Gradient, Gradient descent · p. 7

Cross-validation ML
Train on some folds, test on the held-out fold, rotate, then average.
→ Validation set, Bias–variance tradeoff, Regularization

CUDA INFRA
NVIDIA's platform for programming GPUs.
→ Inference, Prefill, Throughput · p. 6

D

Data drift OPS
Production inputs change from what the system was built and tested on.
→ Model routing, Rate limiting, Eval · p. 21

Glossary A–Z

Continued.

Decision tree ML
Splits data by feature thresholds into a tree of if-then rules.
→ Random forest, Gradient boosting

Decoder-only LLM
Transformer with causal self-attention only. GPT and Llama style.
→ Encoder, Open-weight model, Self-hosted · p. 9

Decoding LLM
Turning probabilities into tokens: greedy, sampling, beam search.
→ Log-probability, Token, Tokenizer · p. 3

Dense retrieval RAG
Search with embedding vectors, matching by meaning.
→ Embedding, Embedding model, Embedding dimension · p. 5

Distillation TRAIN
Training a small student model to imitate a larger teacher.
→ Fine-tuning, LoRA, QLoRA · p. 8

DPO (direct preference optimization) TRAIN
Aligns a model from preference pairs, with no separate reward model or RL loop.
→ Base model, Loss function, Cross-entropy · p. 7

Dropout DL
Randomly zeroing activations during training to reduce overfitting.
→ Cross-validation, Validation set, Bias–variance tradeoff

E

ef_search RAG
HNSW setting: candidates explored per query. Higher is more accurate and slower.
→ Vector database, Matryoshka embedding, Semantic search · p. 5

Embedding MATH
A vector that represents text so similar meanings sit close together.
→ Embedding model, Embedding dimension, Cosine similarity · p. 5

Embedding dimension RAG
Length of the embedding vector, e.g. 384, 1,536, 4,096.
→ Cosine similarity, HNSW, ANN · p. 5

Embedding model RAG
Model that turns text into embedding vectors.
→ Embedding dimension, Cosine similarity, HNSW · p. 5

Encoder LLM
Transformer with bidirectional attention that produces representations (BERT style).
→ Open-weight model, Self-hosted, Reasoning model · p. 9

Epoch TRAIN
One full pass over the training dataset.
→ Batch size, Validation loss, Overfitting · p. 7

Eval (evaluation) OPS
A measured test of quality on a fixed dataset.
→ Canary release, Fallback model, Gateway · p. 21

Evaluator-optimizer AGENT
One LLM generates, another critiques, and the loop repeats until the check passes.
→ Orchestrator-workers, Orchestration, Reflection · p. 16

F

F1 score EVAL
Harmonic mean of precision and recall: 2PR / (P + R).
→ Confusion matrix, ROC-AUC, Precision

Faithfulness EVAL
Share of claims in an answer that the retrieved context supports.
→ Context precision, Context recall, Hallucination · p. 13

Fallback model OPS
Backup model used when the primary fails, times out or is rate-limited.
→ Gateway, LiteLLM, Data drift · p. 21

FastAPI INFRA
Python web framework often used to serve AI backends.

Feature engineering ML
Creating useful input variables from raw data.
→ Dropout, Cross-validation, Validation set

Feed-forward network (FFN) LLM
Per-token network inside each transformer block. Holds a large share of the parameters.
→ Neural network, Parameter, Weights

Few-shot LLM
Showing the model a handful of examples in the prompt.
→ Zero-shot, In-context learning, Context engineering · p. 20

Fine-tuning TRAIN
Further training a pretrained model on task-specific data.
→ LoRA, QLoRA, PEFT · p. 8

FlashAttention INFER
Exact attention computed in tiles to cut memory traffic.
→ Context window, Attention, Self-attention · p. 4

Function calling AGENT
Model returns structured arguments for a function your code then runs. Also called tool calling.
→ Idempotent, Pydantic, Chain-of-thought · p. 14

Glossary A–Z

Continued.

G

Gateway (LLM) OPS
Proxy in front of models: routing, auth, rate limits, logging and fallbacks.
→ LiteLLM, Data drift, Model routing · p. 21

GGUF INFRA
File format for quantized models used by llama.cpp and Ollama.
→ Speculative decoding, GPU, CUDA · p. 6

Golden dataset OPS
Curated inputs with expected outputs, used as ground truth in evals.
→ Recall@k, Precision@k, MRR · p. 13

GPU INFRA
Processor built for massive parallel math. The workhorse of training and inference.
→ CUDA, Inference, Prefill · p. 6

GQA (grouped-query attention) LLM
Query heads share key/value heads in groups, shrinking the KV cache.
→ KV cache, RoPE, Positional encoding · p. 4

Gradient MATH
Vector of partial derivatives showing how the loss changes with each weight.
→ Gradient descent, Backpropagation, Learning rate · p. 7

Gradient boosting ML
Trees built in sequence, each correcting the previous errors (XGBoost, LightGBM).
→ Decision tree, Random forest

Gradient descent MATH
Update rule $\theta \leftarrow \theta - \eta \nabla L$ that moves weights downhill on the loss.
→ Backpropagation, Learning rate, Epoch · p. 7

GraphRAG RAG
RAG over a knowledge graph of entities and relations, with community summaries for corpus-wide questions.
→ Knowledge graph, Grounding, RAG · p. 12

Greedy decoding LLM
Always pick the single most likely next token.
→ Sampling, Beam search, Autoregressive · p. 3

Grounding RAG
Tying answers to provided sources.
→ RAG, Agentic RAG, GraphRAG · p. 12

Guardrails SEC
Checks on inputs and outputs that enforce safety, format and policy.
→ PII, Presidio, Prompt injection · p. 24

H

Hallucination EVAL
Fluent output that is not supported by the source or by reality.
→ RAGAS, Golden dataset, Recall@k · p. 13

Handoff AGENT
Transfer of control from one agent to another, usually via shared state.
→ Agent Skills, A2A, Multi-agent · p. 17

Helicone OPS
Observability gateway for LLM requests.
→ LLM-as-judge, Benchmark, Trace · p. 23

HNSW RAG
Hierarchical Navigable Small World. Layered graph index for fast approximate vector search.
→ ANN, ef_search, Vector database · p. 5

Human-in-the-loop AGENT
A pause where a person approves or edits before the workflow continues.
→ Middleware, Postgres, LangGraph · p. 15

Hybrid search RAG
Combining dense (meaning) and keyword (BM25) results.
→ RRF, Re-ranking, Bi-encoder · p. 11

HyDE RAG
Hypothetical Document Embeddings. Embed an LLM-written draft answer to retrieve better.
→ MMR, Late interaction, SPLADE · p. 11

Hyperparameter TRAIN
A setting chosen before training, such as learning rate or batch size.
→ Feature engineering, Dropout, Cross-validation

I

IDCG EVAL
DCG of a perfect ranking. Used to scale nDCG to 0–1.
→ Faithfulness, Context precision, Context recall · p. 13

Idempotent AGENT
Safe to repeat: running it twice has the same effect as once.
→ Pydantic, Chain-of-thought, Agent · p. 14

In-context learning LLM
Learning a task from examples in the prompt, with no weight updates.
→ Context engineering, Context rot, Compaction · p. 20

Glossary A–Z

Continued.

Inference INFER
Running a trained model to produce outputs.
→ Prefill, Throughput, Continuous batching · p. 6

Instruction tuning TRAIN
Fine-tuning on instruction and response pairs so the model follows directions.
→ Alignment, RLHF, DPO · p. 7

J

Jailbreak SEC
A prompt crafted to bypass a model's safety rules.
→ Lethal trifecta, Guardrails, PII · p. 24

JSON mode LLM
Setting that guarantees valid JSON, but not a particular shape.
→ Constrained decoding, Structured output · p. 22

JSON-RPC AGENT
Lightweight remote-call protocol. The message format used by MCP.
→ MRTR, MCP · p. 19

K

k-means ML
Clustering that assigns points to k centroids and updates the centroids iteratively.
→ PCA, k-NN, Clustering

k-NN (k-nearest neighbours) ML
Predicts or retrieves using the k closest examples.
→ Clustering, k-means, PCA

KL divergence MATH
Measures how one probability distribution differs from another.
→ CNN, RNN and LSTM

Knowledge graph RAG
Entities as nodes and relations as edges.
→ Grounding, RAG, Agentic RAG · p. 12

KV cache INFER
Stored keys and values of earlier tokens, so each decode step computes only the newest token.
→ RoPE, Positional encoding, Sliding-window attention · p. 4

L

Langfuse OPS
Open-source tool for LLM tracing, prompts and evals. Can be self-hosted.
→ Helicone, LLM-as-judge, Benchmark · p. 23

LangGraph AGENT
Library for building stateful agent workflows as graphs.
→ Checkpoint, Super-step, Reducer · p. 15

LangSmith OPS
LangChain's platform for tracing, datasets and evals.
→ Phoenix, Langfuse, Helicone · p. 23

Late interaction (ColBERT) RAG
Keeps one vector per token and scores query-document pairs token by token.
→ SPLADE, BM25, Hybrid search · p. 11

Latency OPS
Time from request to response. See TTFT.
→ p95 latency, TTFT, Streaming · p. 25

Layer normalization DL
Rescales activations for stable training. RMSNorm skips the mean-centering.
→ RMSNorm, Batch normalization, Residual connection

Learning rate TRAIN
Step size η in gradient descent.
→ Epoch, Batch size, Validation loss · p. 7

Lethal trifecta SEC
Private data + untrusted content + a way to send data out. Together they allow data theft through prompt injection. Remove one.
→ Guardrails, PII, Presidio · p. 24

Linear regression ML
Predicts a number as a weighted sum of inputs: y = w·x + b.
→ Logistic regression, Supervised learning, Self-supervised learning

LiteLLM OPS
Open-source gateway and SDK that puts many model APIs behind one interface.
→ Data drift, Model routing, Rate limiting · p. 21

LlamaIndex RAG
Framework for ingesting, indexing and querying data with LLMs.

LLM (large language model) LLM
Neural network trained on very large text corpora to predict the next token.
→ Transformer, Decoder-only, Encoder · p. 9

LLM-as-judge EVAL
Using a model to grade outputs against criteria.
→ Benchmark, Trace, Span · p. 23

Log-probability MATH
log p of a token. Shows model confidence.
→ Token, Tokenizer, BPE · p. 3

Glossary A–Z

Continued.

Logistic regression ML
Linear model passed through a sigmoid to output a class probability.
→ Supervised learning, Self-supervised learning, Classification

Logits MATH
Raw scores a model outputs before softmax.
→ Softmax, Temperature, Top-k · p. 3

LoRA (low-rank adaptation) TRAIN
Fine-tunes by training small matrices B·A added to frozen weights.
→ QLoRA, PEFT, Adapter · p. 8

Loss function ML
Number measuring prediction error, which training minimises.
→ Cross-entropy, Perplexity, Gradient · p. 7

M

Matryoshka embedding RAG
Trained so the first dimensions can be kept alone as a shorter vector.
→ Semantic search, Dense retrieval, Embedding · p. 5

MCP (Model Context Protocol) AGENT
Open standard that connects AI apps to tools, data and prompts through servers.
→ JSON-RPC, MRTR · p. 19

Metadata filtering RAG
Restricting search by fields such as source, date, tenant or permissions.
→ ACL, Qdrant, pgvector · p. 10

Middleware (agent) AGENT
Hooks before or after model and tool calls, e.g. human approval or logging.
→ Postgres, LangGraph, Checkpoint · p. 15

Mixture of experts (MoE) LLM
A router sends each token to a few expert FFNs, so active parameters are fewer than total.
→ LLM, Transformer, Decoder-only · p. 9

MMR (maximal marginal relevance) RAG
Re-selects results to balance relevance and diversity.
→ Late interaction, SPLADE, BM25 · p. 11

Model routing OPS
Sending each request to a model chosen by difficulty or cost.
→ Rate limiting, Eval, Canary release · p. 21

MRR (mean reciprocal rank) EVAL
Average of 1 / rank of the first relevant result.
→ nDCG, IDCG, Faithfulness · p. 13

MRTR (multi round-trip request) AGENT
MCP pattern: server asks for more input in its result; client retries the request with the answers.
→ MCP, JSON-RPC · p. 19

Multi-agent AGENT
Several specialised agents coordinated by a router, supervisor or hierarchy.
→ Subagent, Supervisor, Handoff · p. 17

Multi-head attention LLM
Several attention heads in parallel, each learning different relations.
→ GQA, KV cache, RoPE · p. 4

Multimodal LLM
Handles more than text: images, audio, video.
→ Mixture of experts, LLM, Transformer · p. 9

N

nDCG EVAL
Ranking quality that rewards relevant results placed higher.
→ IDCG, Faithfulness, Context precision · p. 13

Neural network DL
Layers of weights and activations trained to map inputs to outputs.
→ Parameter, Weights, Activation function

O

Observability OPS
Seeing inside a running system through traces, logs and metrics.
→ OpenTelemetry, LangSmith, Phoenix · p. 23

Ollama INFRA
Tool for running open-weight models locally.
→ GGUF, Speculative decoding, GPU · p. 6

Open-weight model LLM
Model whose trained weights can be downloaded. Licences vary.
→ Self-hosted, Reasoning model, Thinking tokens · p. 9

OpenTelemetry (GenAI) OPS
Open tracing standard. Its GenAI conventions define LLM span attributes.
→ LangSmith, Phoenix, Langfuse · p. 23

Orchestration AGENT
Coordinating steps, tools and models in one workflow.
→ Reflection, Prompt chaining, Router · p. 16

Glossary A–Z

Continued.

Orchestrator-workers AGENT
A lead LLM splits a task into subtasks at runtime, sends them to workers, merges results.
→ Orchestration, Reflection, Prompt chaining · p. 16

Overfitting ML
Model memorises training data and generalises poorly.
→ Underfitting, Adam optimizer, Pre-training · p. 7

P

p95 latency OPS
The time 95% of requests finish within. Shows the slow tail that averages hide.
→ TTFT, Streaming, Semantic cache · p. 25

PagedAttention INFER
vLLM technique that stores the KV cache in fixed-size blocks, like OS memory pages.
→ Quantization, vLLM, Ollama · p. 6

Parameter DL
A learned weight. Model size is counted in parameters.
→ Weights, Activation function, ReLU

Parent-child chunking RAG
Embeds small chunks for precise matching, but returns the larger parent section.
→ Contextual retrieval, Metadata filtering, ACL · p. 10

PCA (principal component analysis) ML
Projects data onto the directions of greatest variance.
→ k-NN, Clustering, k-means

PEFT TRAIN
Parameter-efficient fine-tuning: train few parameters, e.g. LoRA.
→ Adapter, Unsloth, Chat template · p. 8

Perplexity MATH
exp(cross-entropy). Lower means the model is less surprised.
→ Gradient, Gradient descent, Backpropagation · p. 7

pgvector RAG
Postgres extension that stores vectors and runs similarity search.
→ Chunking, Chunk overlap, Recursive chunking · p. 10

Phoenix OPS
Arize's open-source tool for LLM tracing and evals.
→ Langfuse, Helicone, LLM-as-judge · p. 23

PII SEC
Personally identifiable information: names, emails, phone numbers.
→ Presidio, Prompt injection, Jailbreak · p. 24

Plan-and-execute AGENT
A planner writes steps, an executor runs them, and the plan may be revised.
→ CodeAct, Sandbox · p. 18

Positional encoding LLM
Gives the model token order, e.g. RoPE.
→ Sliding-window attention, FlashAttention, Context window · p. 4

Postgres INFRA
Relational database. In agent stacks it often stores checkpoints and memory.
→ LangGraph, Checkpoint, Super-step · p. 15

Pre-training TRAIN
Large-scale initial training on next-token prediction.
→ Instruction tuning, Alignment, RLHF · p. 7

Precision EVAL
Share of positive predictions that are correct: TP / (TP + FP).
→ Recall, F1 score, Confusion matrix

Precision@k EVAL
Share of the top k results that are relevant.
→ MRR, nDCG, IDCG · p. 13

Prefill INFER
First pass: the whole prompt is processed in parallel.
→ Throughput, Continuous batching, PagedAttention · p. 6

Presidio SEC
Microsoft's open-source toolkit for detecting and anonymising PII.
→ Prompt injection, Jailbreak, Lethal trifecta · p. 24

Prompt caching OPS
Reuse of an already-processed prompt prefix, so repeat input is cheaper and faster.
→ Few-shot, Zero-shot, In-context learning · p. 20

Prompt chaining AGENT
The output of one LLM call feeds the next, with checks between steps.
→ Router, Evaluator-optimizer, Orchestrator-workers · p. 16

Prompt injection SEC
Malicious text that hijacks a model's instructions, often via retrieved or tool content.
→ Jailbreak, Lethal trifecta, Guardrails · p. 24

Pydantic AGENT
Python library for data validation. Used for tool arguments and structured output.
→ Chain-of-thought, Agent, ReAct · p. 14

REFERENCE

Glossary A–Z

Continued.

Q

Qdrant RAG
Open-source vector database, often self-hosted.
→ pgvector, Chunking, Chunk overlap · p. 10

QLoRA TRAIN
LoRA on top of a 4-bit quantized base model.
→ PEFT, Adapter, Unsloth · p. 8

Quantization INFER
Storing weights in fewer bits (INT8, 4-bit) to cut memory and speed up inference.
→ vLLM, Ollama, GGUF · p. 6

Query rewriting RAG
Having an LLM rephrase the question so retrieval works better.
→ HyDE, MMR, Late interaction · p. 11

R

RAG (retrieval-augmented generation) RAG
Retrieve relevant documents, then have the model answer using them.
→ Agentic RAG, GraphRAG, Knowledge graph · p. 12

RAGAS EVAL
Open-source framework of RAG metrics.
→ Golden dataset, Recall@k, Precision@k · p. 13

Random forest ML
Many decision trees trained on random samples, averaged for a stable prediction.
→ Gradient boosting, Decision tree

Rate limiting OPS
Capping requests or tokens per time window.
→ Eval, Canary release, Fallback model · p. 21

Re-ranking RAG
Re-scoring the top retrieved results with a more accurate model.
→ Bi-encoder, Cross-encoder, Query rewriting · p. 11

ReAct AGENT
Reason plus Act. A loop of Thought, Action, Observation.
→ Tool calling, Function calling, Idempotent · p. 14

Reasoning model LLM
Model trained to think before answering, using thinking tokens.
→ Thinking tokens, Multimodal, Mixture of experts · p. 9

Recall EVAL
Share of actual positives that the model finds: TP / (TP + FN).
→ F1 score, Confusion matrix, ROC-AUC

Recall@k EVAL
Share of all relevant documents found in the top k results.
→ Precision@k, MRR, nDCG · p. 13

Recursive chunking RAG
Splits by headings, then paragraphs, then sentences until pieces fit.
→ Semantic chunking, Parent-child chunking, Contextual retrieval · p. 10

Reducer AGENT
Function that merges a node's partial update into state, e.g. add_messages appends.
→ thread_id, Human-in-the-loop, Middleware · p. 15

Reflection AGENT
An agent critiques and revises its own output.
→ Prompt chaining, Router, Evaluator-optimizer · p. 16

Regularization ML
Techniques that reduce overfitting, e.g. weight decay and dropout.
→ Hyperparameter, Feature engineering, Dropout

ReLU DL
max(0, x). Simple, fast activation.
→ Sigmoid, SwiGLU, Feed-forward network

Residual connection DL
Adds a layer's input to its output, easing deep training.
→ Layer normalization, RMSNorm, Batch normalization

RLHF TRAIN
Reinforcement learning from human feedback. Learn a reward from rankings, then optimise the model.
→ DPO, Base model, Loss function · p. 7

RMSNorm DL
Normalises by root-mean-square: x / sqrt(mean(x^2) + ε) · g.
→ Batch normalization, Residual connection, Layer normalization

RNN and LSTM DL
Sequence models read step by step. LSTM adds gates for longer memory. Largely replaced by transformers.
→ KL divergence, CNN

ROC-AUC EVAL
Area under the true-positive vs false-positive curve. 0.5 is random, 1.0 is perfect.
→ Precision, Recall, F1 score

RoPE (rotary position embedding) LLM
Encodes position by rotating Q and K vectors.
→ Positional encoding, Sliding-window attention, FlashAttention · p. 4

Router AGENT
Component that picks which path, agent or model handles an input.
→ Evaluator-optimizer, Orchestrator-workers, Orchestration · p. 16

Glossary A–Z

Continued.

RRF (reciprocal rank fusion) RAG
Merges ranked lists: score = Σ 1 / (k + rank).
→ Re-ranking, Bi-encoder, Cross-encoder · p. 11

S

Sampling LLM
Drawing the next token at random from the probabilities.
→ Beam search, Autoregressive, Decoding · p. 3

Sandbox SEC
An isolated environment (container, no network) where untrusted or generated code runs safely.
→ Plan-and-execute, CodeAct · p. 18

Self-attention LLM
Attention where Q, K and V all come from the same sequence.
→ Multi-head attention, GQA, KV cache · p. 4

Self-hosted INFRA
Running models and services on your own servers instead of a vendor API.
→ Reasoning model, Thinking tokens, Multimodal · p. 9

Self-supervised learning ML
Learning from labels hidden in the data itself, like the next word. LLM pre-training works this way.
→ Classification, Linear regression, Logistic regression

Semantic cache OPS
Reuses answers for questions with similar meaning.
→ Latency, p95 latency, TTFT · p. 25

Semantic chunking RAG
Splits text where the meaning changes.
→ Parent-child chunking, Contextual retrieval, Metadata filtering · p. 10

Semantic search RAG
Search by meaning rather than exact words.
→ Dense retrieval, Embedding, Embedding model · p. 5

Sigmoid DL
Squashes any value into (0, 1): 1 / (1 + e^(−x)).
→ SwiGLU, Feed-forward network, Neural network

Sliding-window attention LLM
Each token attends only to a recent window, cutting cost.
→ FlashAttention, Context window, Attention · p. 4

Softmax MATH
Turns scores into probabilities that sum to 1: exp(z_i) / Σ exp(z_j).
→ Temperature, Top-k, Top-p · p. 3

Span OPS
One timed step inside a trace.
→ Observability, OpenTelemetry, LangSmith · p. 23

Speculative decoding INFER
A small model drafts tokens and the large model verifies them in one pass.
→ GPU, CUDA, Inference · p. 6

SPLADE RAG
Learned sparse retrieval: keyword-style vectors with weights learned by a model.
→ BM25, Hybrid search, RRF · p. 11

Streaming OPS
Sending tokens to the user as they are generated, so the answer appears sooner.
→ Semantic cache, Latency, p95 latency · p. 25

Structured output LLM
Constraining output to a JSON schema.
→ JSON mode, Constrained decoding · p. 22

Subagent AGENT
An agent invoked as a tool by a main agent, working in its own context.
→ Supervisor, Handoff, Agent Skills · p. 17

Super-step AGENT
One LangGraph tick: all nodes scheduled together run, then state is checkpointed.
→ Reducer, thread_id, Human-in-the-loop · p. 15

Supervised learning ML
Learning from labelled examples.
→ Self-supervised learning, Classification, Linear regression

Supervisor AGENT
Agent that delegates to workers and merges their results.
→ Handoff, Agent Skills, A2A · p. 17

SwiGLU DL
Gated feed-forward activation used in many modern LLMs.
→ Feed-forward network, Neural network, Parameter

System prompt LLM
Developer instructions that frame the model's behaviour.
→ Prompt caching, Few-shot, Zero-shot · p. 20

T

Temperature LLM
Divides logits before softmax. Low is predictable, high is varied.
→ Top-k, Top-p, Greedy decoding · p. 3

Glossary A–Z

Continued.

Thinking tokens LLM
Hidden reasoning tokens generated before the answer. Billed as output tokens.
→ Multimodal, Mixture of experts, LLM · p. 9

thread_id AGENT
ID that groups the checkpoints of one conversation. Like a session ID.
→ Human-in-the-loop, Middleware, Postgres · p. 15

Throughput OPS
Requests or tokens processed per second.
→ Continuous batching, PagedAttention, Quantization · p. 6

Token LLM
Unit of text a model reads and writes, often a piece of a word.
→ Tokenizer, BPE, Vocabulary · p. 3

Tokenizer LLM
Converts text to token IDs and back.
→ BPE, Vocabulary, Logits · p. 3

Tool calling AGENT
See Function calling.
→ Function calling, Idempotent, Pydantic · p. 14

Top-k LLM
Sample only from the k most likely tokens.
→ Top-p, Greedy decoding, Sampling · p. 3

Top-p (nucleus) LLM
Sample only from the smallest set of tokens whose probabilities sum to p.
→ Greedy decoding, Sampling, Beam search · p. 3

Trace OPS
End-to-end record of one request as a tree of spans.
→ Span, Observability, OpenTelemetry · p. 23

Transformer LLM
Architecture built from attention and feed-forward blocks (2017).
→ Decoder-only, Encoder, Open-weight model · p. 9

TTFT (time to first token) OPS
Delay before the first output token appears.
→ Streaming, Semantic cache, Latency · p. 25

U

Underfitting ML
Model too simple to capture the pattern, even on training data.
→ Adam optimizer, Pre-training, Instruction tuning · p. 7

Unsloth TRAIN
Library for faster, lower-memory LoRA and QLoRA fine-tuning.
→ Chat template, Distillation, Fine-tuning · p. 8

V

Validation loss TRAIN
Loss on held-out data. Rising while training loss falls means overfitting.
→ Overfitting, Underfitting, Adam optimizer · p. 7

Validation set ML
Held-out data used to tune choices, kept separate from the test set.
→ Bias–variance tradeoff, Regularization, Hyperparameter

Vector database RAG
Database optimised to store embeddings and run similarity search.
→ Matryoshka embedding, Semantic search, Dense retrieval · p. 5

vLLM INFER
High-throughput inference engine with PagedAttention and continuous batching.
→ Ollama, GGUF, Speculative decoding · p. 6

Vocabulary LLM
The set of all tokens a model knows.
→ Logits, Softmax, Temperature · p. 3

W

Weights DL
The learned numbers in a model. See Parameter.
→ Activation function, ReLU, Sigmoid

Z

Zero-shot LLM
Asking a model to do a task with no examples.
→ In-context learning, Context engineering, Context rot · p. 20

ABOUT THE AUTHOR

Raza Jahangir

AI Systems Engineer

Backend engineer since 2012. I build production-ready AI agents and RAG systems, and private AI stacks for businesses.

`razajahangir.dev`

© 2026 Raza Jahangir · Edition 1, November 2026
Free to share unchanged, with credit to razajahangir.dev. Not for resale.
Product and framework names belong to their owners. Check details against current documentation.