FREE · LEARN.AIWITHPRIYABRAT.COM

AI Agents vs RAG Comparison Guide

When to retrieve and answer, when to let an AI agent plan and act, and how to combine them: differences, a decision tree, cost and latency, agentic RAG patterns, evaluation and guardrails.

By **Priyabrat Dalbehera** · GenAI engineer

3 pages | **5** parts | **1** decision tree | **5.6×** agent cost, worked example

What's inside

1 **The difference** p. 2
definitions, the RAG pipeline, the agent loop, the spectrum

2 **When to use which** p. 2
decision tree, 7 use cases, cost and latency worked out

3 **Agentic RAG & hybrids** p. 3
retrieval as a tool, routing, corrective RAG, a minimal agent loop

4 **Evals, guardrails & security** p. 3
metrics, 7 failure modes, guardrails, threat model

5 **Tools & interview** p. 3
frameworks, 6 quick answers

How to read it
Each box compares RAG and agents on one question, with a clear recommendation.

Tip & Watch out
Green boxes are good habits. Amber boxes are traps that cost money or trust.

Prices are examples
Cost figures use placeholder prices to show the method; use your provider's prices.

learn.aiwithpriyabrat.com

© 2026 Priyabrat Dalbehera · Free to download and share unchanged, with credit · Not for resale

AI Agents vs RAG Comparison Guide

A/R

Parts 1–2 · definitions · the RAG pipeline · the agent loop · key differences · myths · when to use which · cost

1.1 One-line definitions

RAG	retrieve relevant text, then let the LLM answer from it. **Your code** decides the steps.
AI agent	an LLM in a loop that chooses tools and next steps until the goal is met. **The model** decides.
Agentic RAG	an agent that uses retrieval as one of its tools (Part 3).

1.2 The RAG pipeline

```
question
  → embed / rewrite query
  → search the index (vector, BM25 or hybrid)
  → top 5–10 chunks (+ rerank)
  → prompt: "answer only from these sources"
  → answer with citations              1 LLM call
```

1.3 The agent loop

```
goal
  → LLM: think, pick a tool + arguments
  → run the tool (search, SQL, API, code…)
  → add the result to the context
  → repeat until done or a limit is hit
  → final answer or action             3–20 LLM calls
```

1.4 The key differences

	RAG	Agent
Who decides the steps	your code, fixed	the model, at run time
LLM calls per request	usually 1	several to many
Latency	seconds	seconds to minutes
Cost per request	low, predictable	higher, varies
Can take actions	no: it answers	yes: tools can write, book, send
Predictability	high, easy to test	lower: needs limits and evals
Main failure	wrong or missing chunks	wrong tool, loops, runaway cost

1.5 It's a spectrum, not a choice of two

1	**Plain RAG**	retrieve → answer
2	**Advanced RAG**	+ query rewriting, hybrid search, reranking
3	**Workflow / chain**	fixed steps in code: classify → retrieve → draft → check
4	**Single agent**	the model chooses tools in a loop
5	**Multi-agent**	several agents with roles hand work to each other

Tip: start at the lowest level that solves the problem. Move right only when an eval shows you must.

1.6 Myths, corrected

"Agents replace RAG"	agents usually **use** RAG as a tool for facts
"Agents are always smarter"	on fixed tasks they're slower, costlier and less reliable
"RAG can't do multi-step"	query rewriting and multi-hop chains handle a lot before you need an agent
"Bigger context makes RAG pointless"	retrieval is still cheaper, faster and cites its sources

2.1 Which one? Ask these questions

```
Is the answer in your documents?
 ├─ yes → does one search find it?
 │        ├─ yes → RAG
 │        └─ no (multi-hop, compare, combine)
 │                 → agentic RAG
 └─ no → does it need live data or actions
         (APIs, SQL, booking, sending)?
          ├─ steps always the same → workflow
          └─ steps depend on results → agent
```

2.2 Use cases

Task	Best fit
HR policy or product FAQ bot	RAG
"Compare our 2024 and 2025 contracts"	agentic RAG (several searches)
Support bot that checks orders and starts returns	agent with tools + RAG for policy
Invoice processing: extract → validate → post	workflow (fixed steps)
Questions over a SQL database	text-to-SQL tool, often in an agent
Research report from the web and internal docs	agent (or multi-agent)
Summarise one long document	neither: map-reduce prompt chain

2.3 Choose RAG when…

- ☐ answers live in documents you control
- ☐ you need citations and an audit trail
- ☐ latency must be a few seconds
- ☐ cost per question must stay low and predictable
- ☐ nothing should be changed in other systems

2.4 Choose an agent when…

- ☐ the steps depend on what earlier steps return
- ☐ it must call APIs, run SQL or take actions
- ☐ it must combine several sources or tools
- ☐ users accept a slower answer for a harder task
- ☐ you can add limits, approvals and evals

2.5 Worked example: cost per request

	Tokens	Cost
RAG	1 call: 4,000 in + 300 out	$0.0130
Agent	5 calls, context grows 3k → 7k, 200 out each	$0.0725

Same model and prices (placeholders: $2.50 / $10.00 per 1M tokens). The agent costs about **5.6×** more per request, mostly because every step re-sends the growing context.

2.6 Latency, roughly

	What happens	Typical
RAG	search ~0.2 s + one LLM call ~1–3 s	≈ 1–3 s
Agent	5 LLM calls + 4 tool calls in sequence	≈ 8–20 s

Tip: stream partial progress ("Searching contracts…") so a slow agent still feels responsive.

learn.aiwithpriyabrat.com

3.1 Agentic RAG: retrieval as a tool

```
TOOLS = [{
  "name": "search_docs",
  "description": "Search the company handbook. Use for "
                 "policy questions. Returns numbered chunks.",
  "parameters": {"type": "object",
                 "properties": {"query": {"type": "string"},
                                "top_k": {"type": "integer"}},
                 "required": ["query"]},
}]
# the agent can search several times, rephrase,
# and combine results before answering
```

3.2 Patterns between RAG and agents

Query routing	pick the right index or tool per question (HR docs, product docs, SQL)
Corrective RAG	grade the retrieved chunks; if they're poor, rewrite the query or search the web
Self-RAG	the model decides whether to retrieve and checks its own answer against sources
Multi-hop	answer part 1, use it to search for part 2 ("who manages the team that owns X?")

3.3 Hybrid architecture that works in production

```
user → router (small model)
   ├─ FAQ / policy      → RAG pipeline        (fast, cheap)
   ├─ account actions   → agent + tools       (verified user)
   │                        + RAG for policy
   └─ unclear / risky   → human hand-off
```

3.4 A minimal agent loop

```
def run_agent(goal, tools, max_steps=8):
    messages = [{"role": "system", "content": RULES},
                {"role": "user", "content": goal}]
    for _ in range(max_steps):
        reply = llm(messages, tools=tools)
        if not reply.tool_calls:          # done
            return reply.content
        messages.append(reply)
        for call in reply.tool_calls:
            result = TOOLS[call.name](**call.args)
            messages.append(tool_result(call.id, result))
    return "Stopped: step limit reached."  # never loop forever
```

4.1 Evaluate them differently

	RAG	Agent
Retrieval	context precision and recall: did we find the right chunks?	tool-call accuracy: right tool, right arguments?
Answer	faithfulness, answer relevancy, citation accuracy	task success rate: was the goal achieved?
Efficiency	latency, tokens per answer	steps per task, cost per task, loop rate
Safety	injection via documents	unsafe or unapproved actions

4.2 How each one fails

Failure	Where	Fix
Right answer not in the index	RAG	ingestion checks, coverage evals
Wrong chunks retrieved	RAG	hybrid search, reranking, better chunking
Answer not supported by sources	both	grounding prompt, citation checks
Loops or repeats a tool	agent	max steps, detect repeated calls
Wrong tool or arguments	agent	clear tool descriptions, schemas, evals
Runaway cost	agent	token and step budgets per task
Harmful action	agent	least privilege, human approval

4.3 Guardrails every agent needs

- ☐ Max steps and a token/cost budget per task
- ☐ Timeouts on every tool call
- ☐ Tools scoped to the signed-in user (least privilege)
- ☐ Human approval for payments, emails, deletes
- ☐ Log every step: thought, tool, arguments, result
- ☐ A fallback: hand off to a human or plain RAG

4.4 Security: what can go wrong

	RAG	Agent
Prompt injection in retrieved text	wrong or manipulated answer	manipulated **actions**: emails sent, data leaked
Access control	users see chunks they shouldn't: filter by permission before retrieval	tools act beyond the user's rights: scope tools to the session user
Blast radius	one bad answer	a chain of real changes

Watch out: an agent that reads untrusted text **and** can act is the riskiest combination: require approval for anything irreversible.

5.1 Tools you'll meet

LlamaIndex, LangChain	RAG pipelines: loaders, chunking, retrievers, query engines
LangGraph	agents and workflows as graphs, with state and human-in-the-loop
OpenAI Agents SDK, CrewAI	agents with tools and hand-offs; multi-agent setups
Vector DBs: pgvector, Qdrant, Pinecone	the retrieval layer for both
RAGAS, Langfuse, LangSmith	evaluation and tracing

5.2 Quick interview answers

RAG vs agent in one line	RAG retrieves then answers in fixed steps; an agent decides its own steps and can act
Is RAG an agent?	no: no loop, no decisions. With retrieval as a tool inside a loop, it becomes agentic RAG
When not to use an agent	when the steps are known: a workflow is cheaper, faster and easier to test
Why agents cost more	many LLM calls, and each re-sends the growing context
Make agents reliable	small toolset, clear schemas, step and cost limits, approvals, evals
RAG vs fine-tuning	RAG adds fresh facts with citations; fine-tuning changes style or skills

5.3 Keep learning, free

- Notes: RAG (8 topics) and AI Agents (9 topics)
- Prompt Engineering Cheat Sheet
- RAG Chatbot Project Template

learn.aiwithpriyabrat.com

learn.aiwithpriyabrat.com