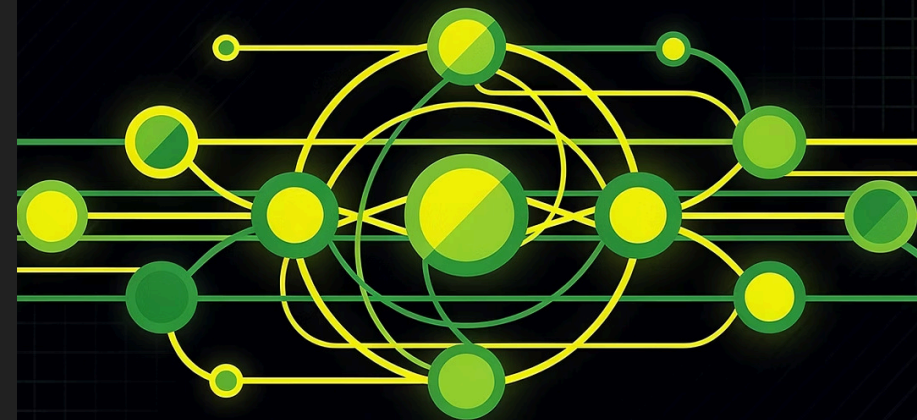


RAGs vs Agenti

Opýtajte sa LLM na dáta vašej spoločnosti a bude hádať. Dva vzory, ktoré to riešia, sú RAG a agenti – a každý rieši iný problém.

BYTEBYTEGO · 23. MÁJA 2026

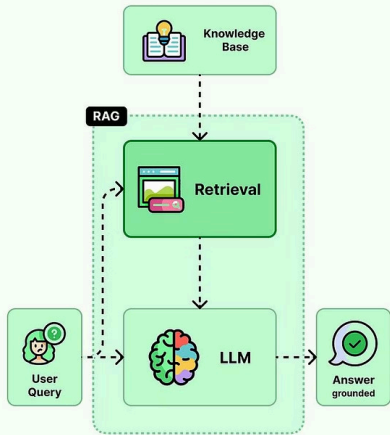


RAG

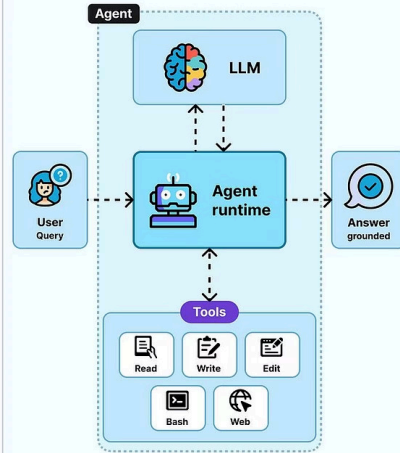
vs

Agent

Retrieve relevant context,
then generate.



Reason, call tools,
take action.

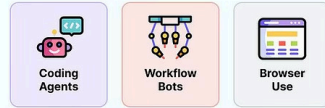


Best when:

Answer lives in
your documents.

Task requires actions
on other systems.

Common use-cases:



RAGs vs Agenti: Prehľad

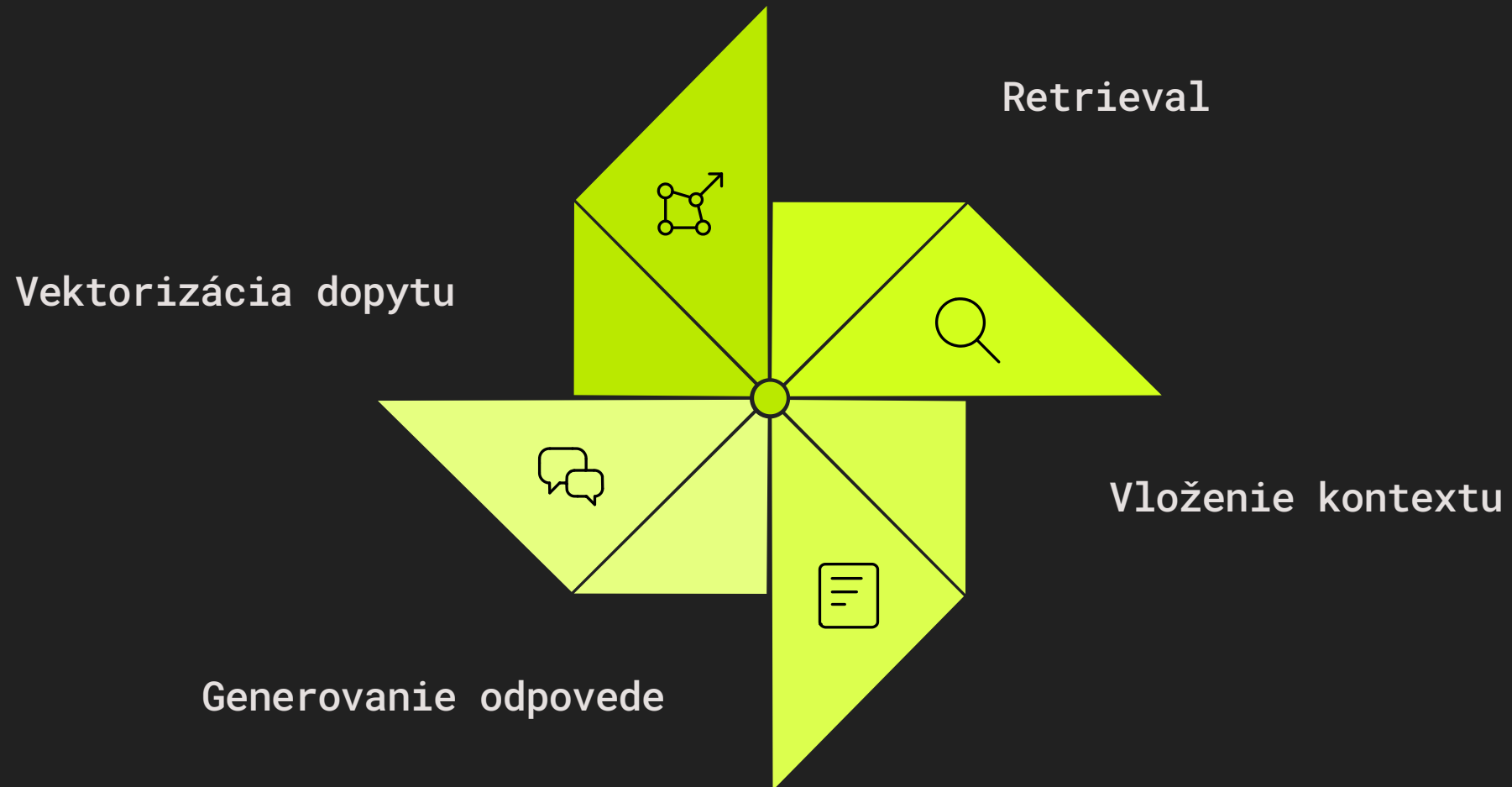
Kedy použiť RAG

Keď odpoveď žije vo vašich dokumentoch – doc Q&A, podnikové vyhľadávanie, znalostná báza podpory.

Kedy použiť Agentu

Keď úloha vyžaduje akcie na iných systémoch – kódovacie agenty, workflow boty, ovládanie prehliadača.

Ako funguje RAG



Jedno získanie. Jedna generácia. **Lacné, predvídateľné a ľahko laditeľné.** RAG kombinuje LLM s retrieval krokom, aby zakotvil odpovede v reálnych dokumentoch.

Ako fungujú Agenti

Príjem dopytu

LLM vyberie
nástroj



LLM uvažuje a
opakuje

Runtime
vykoná
nástroj

Flexibilnejší. Viac tokenov. **Ťažšie ladiť**, pretože chyby sa prenášajú naprieč krokmi. Agenti obalujú LLM do reasoning loopu s nástrojmi na vykonávanie akcií.

Build with Claude Code: Nový Cohort

Spúšťame nový 2-dňový intenzívny kurz postavený na cohortoch — *Build with Claude Code*, ktorý vyučuje John Kim, tréner stoviek inžinierov v Meta. Kurz začína **28. mája 2026**.

Agentic Loop

Context engineering a pamäťové vrstvy pre reálne projekty

MCPs & Hooks

Nástroje a feedback looky pre sebakorekciu Claude Code

Paralelný vývoj

Git worktrees, subagenty a agent tímy

Capstone projekt

Odošlite niečo reálne na vlastnom stacku

Build With Claude Code

2-Day Intensive · Cohort-Based Course · Next Cohort: May 28-29

Session 1

Thursday, May 28, 2026 5-8 PM PT Live

Claude Code Fundamentals

Claude Code Basics

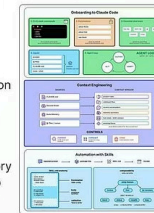
- Agentic loop: gather → act → verify
- Installation and first interactive session
- Slash commands, shortcuts, permissions

Context Engineering

- Treat context as finite working memory
- Use CLAUDE.md, Second Brain, Auto Memory

Automation with Skills

- Reusable workflows and SKILL.md anatomy



Assignments

Self-paced Recorded explain + solution included

Practice the Foundations

Onboard to a New Codebase

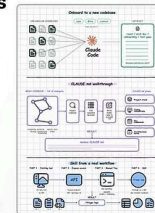
- Explore an unfamiliar repo in plan mode
- Produce a roast, architecture doc, onboarding guide, or test gap analysis

Verification

- Initialize CLAUDE.md and write canonical patterns

Skill from a Real Workflow

- Build endpoint, report template, and / triage-logs skill



Session 2

Friday, May 29, 2026 5-8 PM PT Live

Scaling Up with MCPs, Parallel Agents, and Agentic Engineering

MCPs, CLIs, and Agentic Tooling

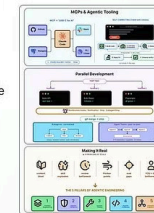
- Install MCP and learn when not to use it
- Browser automation with /chrome
- Self-correcting chain: build → screenshot → fix → verify

Parallel Development

- Git worktrees and multi-agent workflow
- Subagents vs. Agent Teams + hooks

Agentic Engineering

- Audit AI-readiness and design validation harnesses



Assignments

Self-paced Recorded explain + solution included

Build at Scale

Connect Claude to the Real World

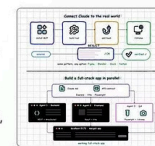
- Install MCP servers: Figma, Blender, Slack, or Notion
- Build a tool that reads/writes through MCP

Build a Full-Stack App in Parallel

- Shared CLAUDE.md with stack, ports, and API contract
- Run Claude instances in separate worktrees

The Full Loop

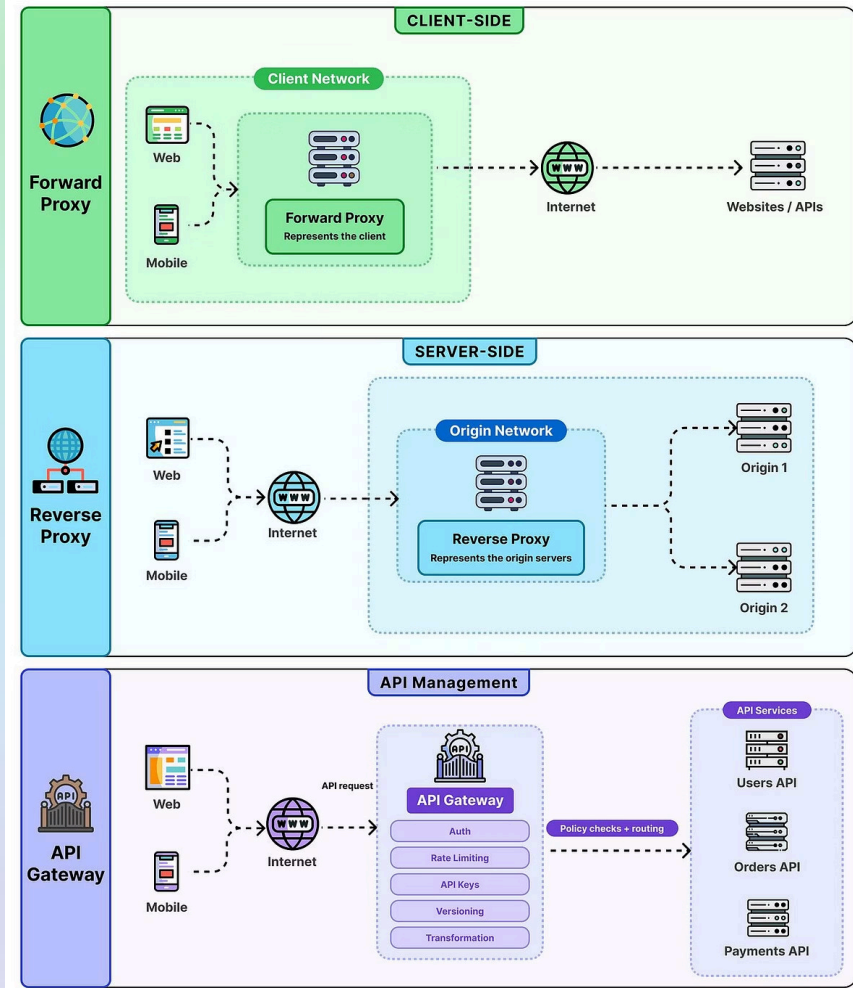
- Reusable skills, validation hooks, tests, and self-correction



Forward Proxy, Reverse Proxy a API Gateway

Ľudia ich často zamieňajú – všetky sedia medzi klientom a serverom. Skutočný rozdiel je, ktorú stranu reprezentujú.

Forward Proxy, Reverse Proxy, and API Gateway Explained



Tri proxy vrstvy vysvetlené

Forward Proxy – strana klienta

Sedí vedľa klienta. Preposiela požiadavky von, cieľ nevidí vašu skutočnú IP. Firemné siete ho používajú na presadzovanie politík, blokovanie stránok a cachovanie prevádzky.

Reverse Proxy – strana servera

Sedí vedľa servera. Klient netuší, koľko strojov je za ním. Proxy rozhoduje, kto spracuje požiadavku, ukončuje TLS a drží backend mimo verejného internetu. Typicky NGINX alebo HAProxy.

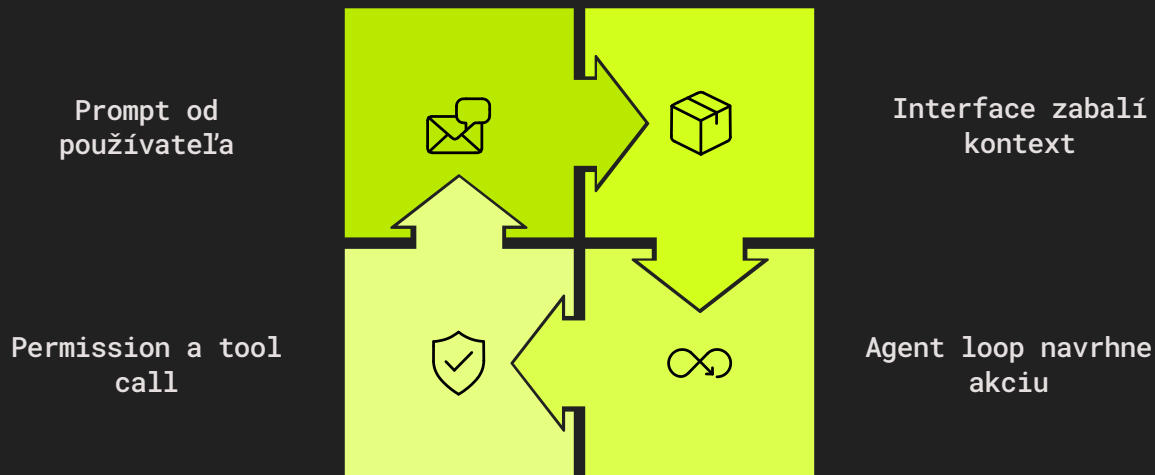
API Gateway – viac než len routing

Reverse proxy, ktorý robí viac: auth, rate limity, API kľúče, verziovanie a tvarovanie požiadaviek. Bez neho musí každá mikroslužba implementovať vlastnú validáciu a throttling.

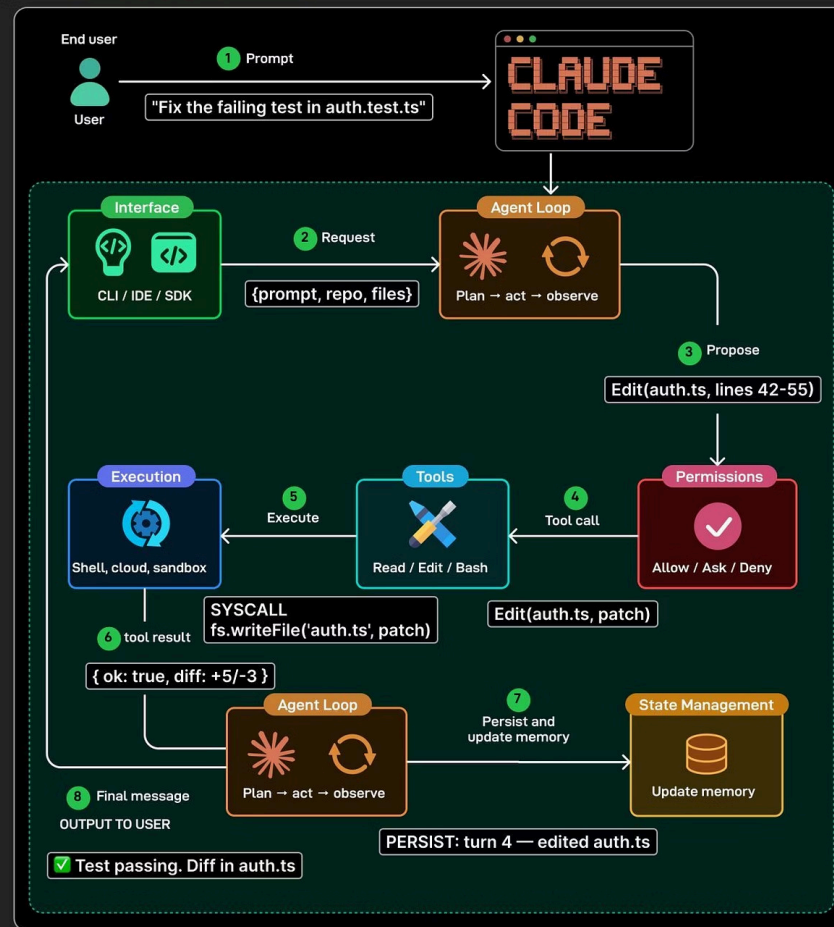
 V reálnych systémoch bežia všetky tri na rôznych vrstvách súčasne.

Ako požiadavka cestuje cez Claude Code

Sledujme reálnu požiadavku: "Oprav zlyhávajúci test v auth.test.ts." Celý systém je len tento loop, opakovaný, kým model neprestane žiadať nástroje.



What Happens When You Hit Enter in Claude code ByteByteGo



5 stratégií kompakcie kontextu v Claude Code

Strategy	What it does	What is compacted	Cost to run
1 Budget Reduction	Replaces oversized tool outputs with a reference	BEFORE: System prompt, CLAUDE.md, History, Tool result - 200KB log AFTER: System prompt, CLAUDE.md, History, ref://result-7d2 Cap	1 / 5 - Size cap
2 Snip	Drops the oldest history segments	BEFORE: System, Turn 1 - explore, Turn 2 - read file, Turn 3 - run tests, Turn 4 - current AFTER: System, Turn 1 - explore, Turn 2 - read file, Turn 3 - run tests, Turn 4 - current Trim	2 / 5 - Trim
3 Micro compact	Fine-grained, cache-aware compression of tool turns	BEFORE: tool_use #n1 - ls, tool_use #n2 - grep result, tool_use #n3 - cat file, tool_use #n4 - diff, Turn N - current AFTER: tool_use #n1 - pruned, tool_use #n2 - pruned, tool_use #n3 - cat file, tool_use #n4 - diff, Turn N - current Compress	3 / 5 - Id-keyed
4 Context collapse	Virtual projection over history. stored REPL untouched	STORED REPL: Turn 1, Turn 2, Turn 3, Turn 4, Turn 5 MODEL SEES: Collapsed view - turns 1-3, Turn 4, Turn 5 Project	4 / 5 - Projection
5 Auto-compact	Full model-generated summary of prior turns	BEFORE: Turn 1 - plan, Turn 2 - explore, Turn 3 - edit, Turn 4 - tests, Turn 5 - current AFTER: [boundary], LLM summary - turns 1-4, Turn 5 - current Summarize	5 / 5 - LLM call

Claude Code používa 5 stratégií spúšťaných v poradí pred každým volaním modelu. Každá sa spustí len ak predchádzajúca nevoľní dostatok miesta – *lazy degradation*.

01

Budget Reduction

Obmedzí veľké výstupy nástrojov – nahradí ich odkazom na obsah.

02

Snip

Ostrihá najstaršie segmenty histórie a vloží hraničný marker.

03

Microcompact

Prune tool turns podľa tool_use_id, aby prompt cache zostala teplá.

04

Context Collapse

Read-time projekcia nad celou históriou – uložené REPLY sa nedotknú.

05

Auto-compact

Posledná záchrana: model vytvorí úplné zhrnutie predchádzajúcich turnov.

Kľúčové závery

RAG vs Agent

Použite RAG, keď odpoveď žije v dokumentoch. Použite agenta, keď úloha vyžaduje akcie na iných systémoch.

Proxy vrstvy

Forward proxy = klient. Reverse proxy = server. API Gateway = reverse proxy + auth, rate limiting a politiky.

Claude Code

Požiadavky prechádzajú 8-krokovým agent loopom. Kontext sa spravuje 5 stratégiami lazy degradácie.

📖 Ďalšie čítanie: **Load Balancer vs API Gateway a Git Workflow: Essential Commands** — ByteByteGo newsletter.

