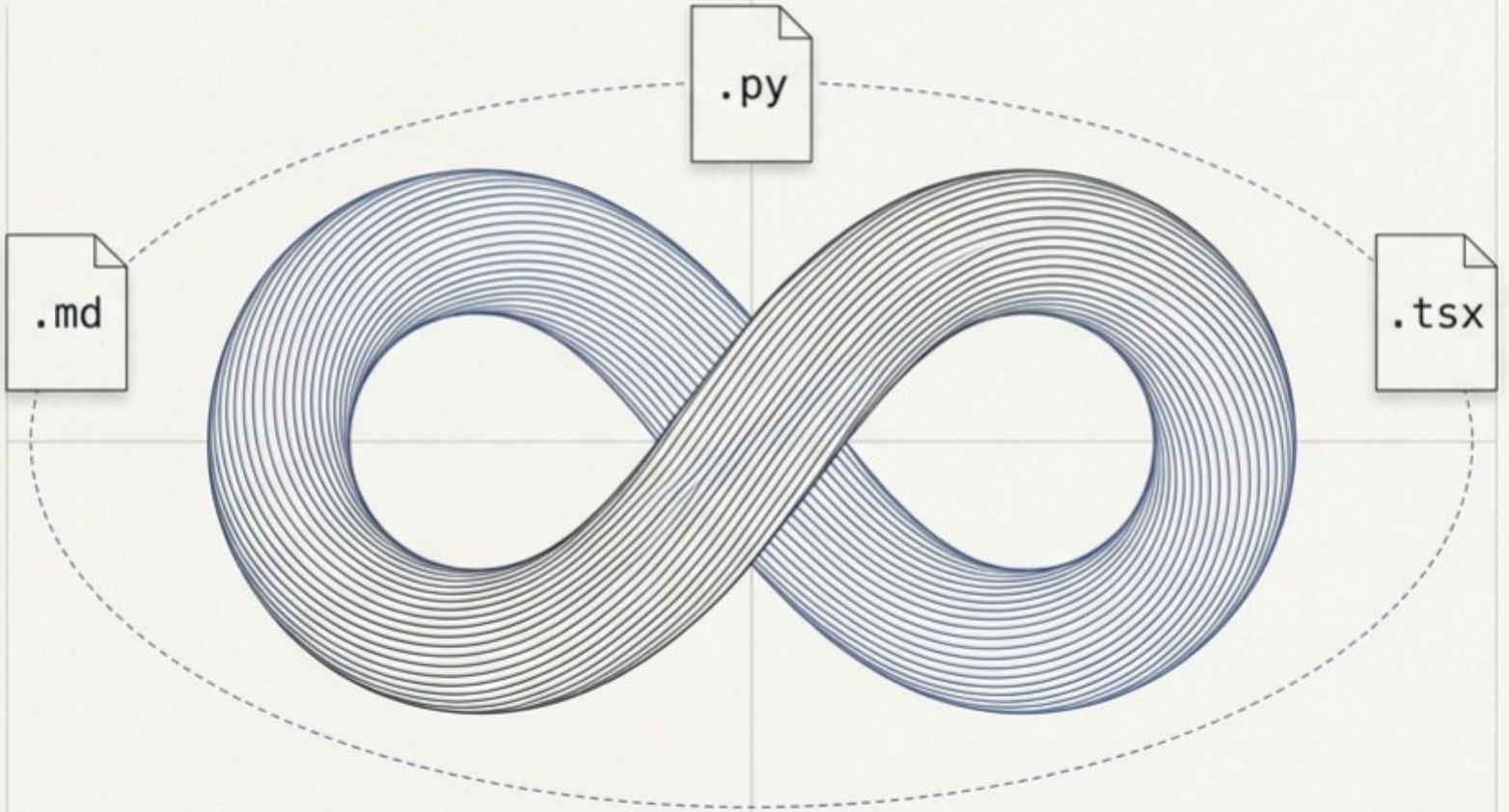
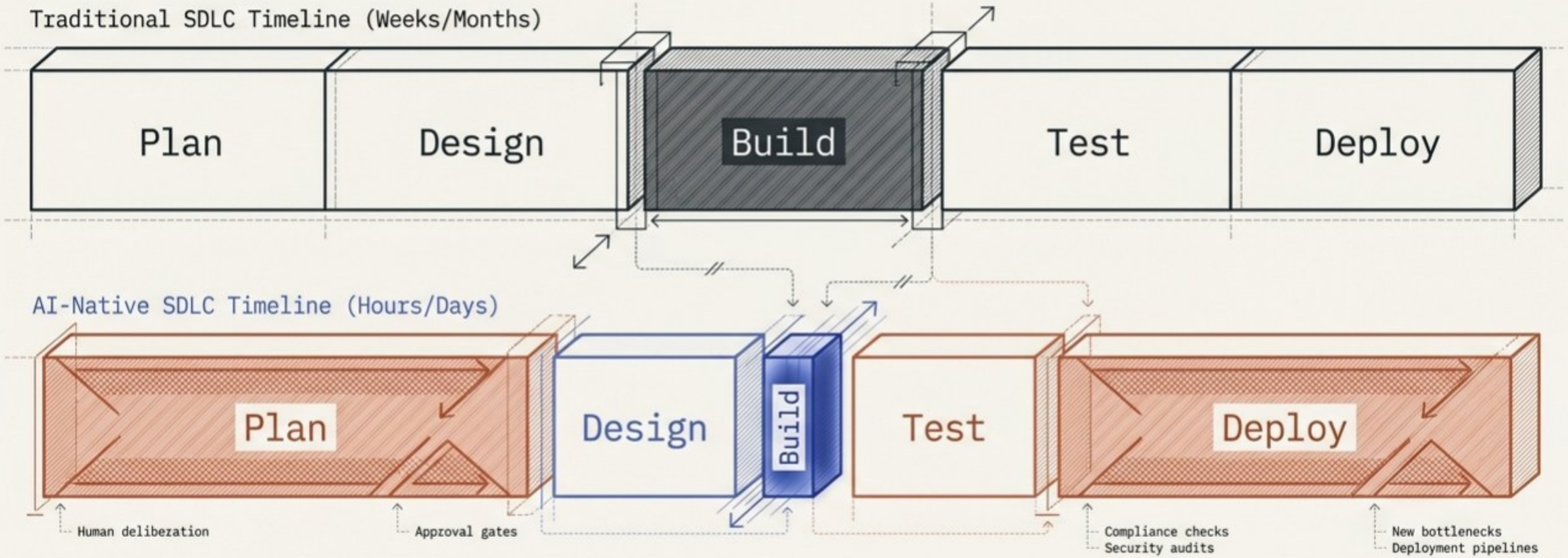
The AI-Native SDLC Blueprint

Transforming the software development lifecycle from human-speed bottlenecks to autonomous continuous loops.

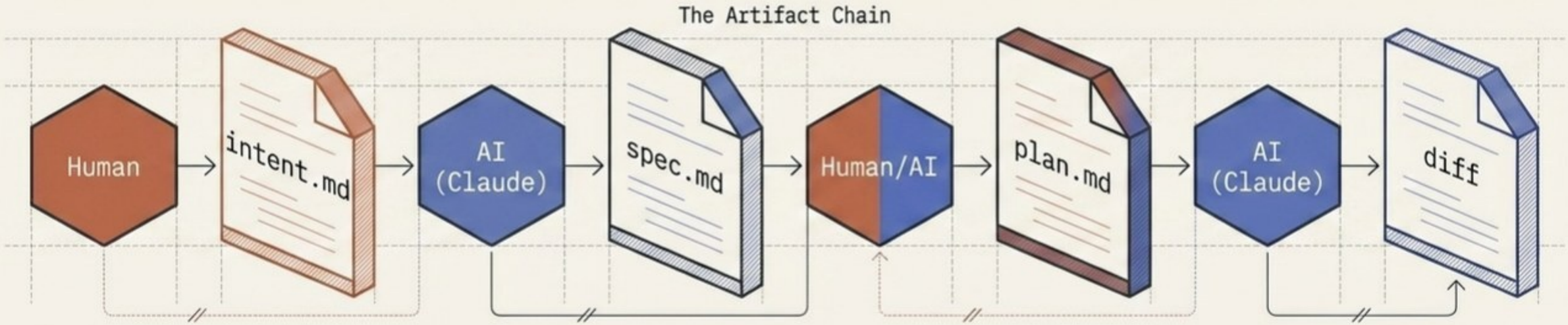


Code is no longer the bottleneck.



When agentic AI collapses the **Build** phase from weeks to hours, **human-speed approval gates** and **planning processes** stall productivity gains. The bottleneck has moved left and right.

The new connective tissue: Markdown.



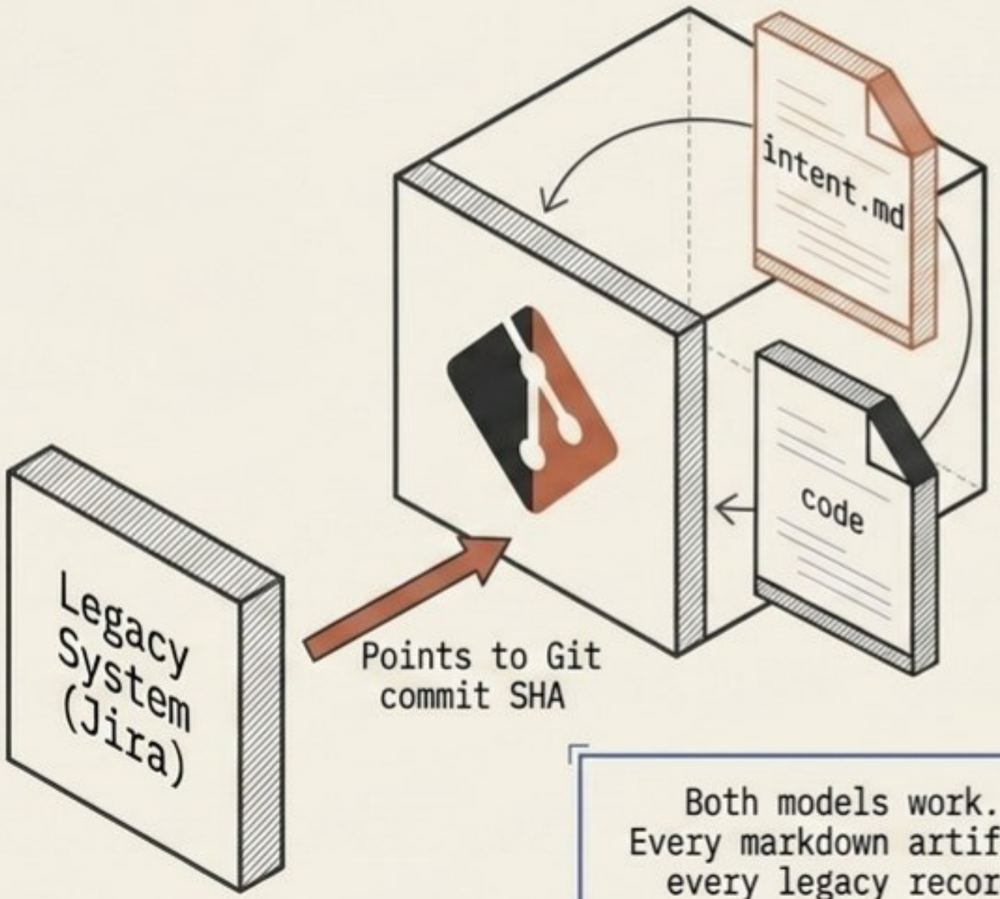
In an AI-Native SDLC, work does not move through meetings or tickets. Every stage ends by **committing a human-readable, machine-actionable artifact** to version control. The chain of commits becomes the audit trail.

The Master SDLC Transformation

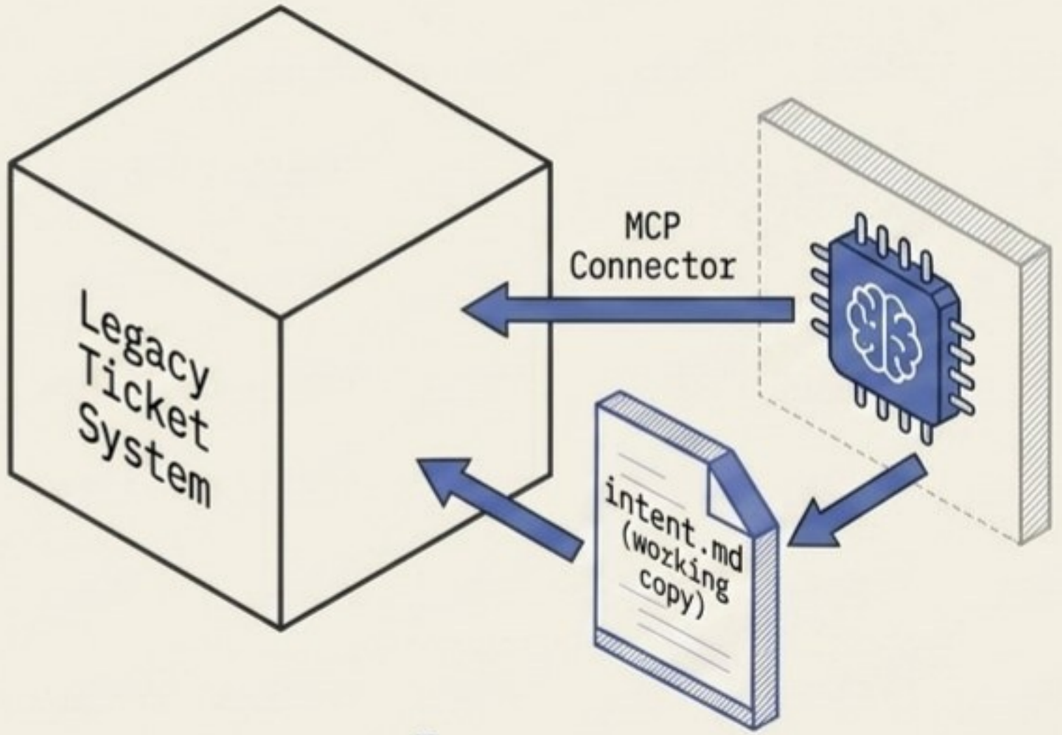
Stage	Traditional SDLC	AI-Native SDLC	Core Artifact
Plan	Requirements gathered by committee	Claude synthesizes pain points straight from the source	intent.md
Design	Specs written by analysts, parsed by designers	Requirements and design compressed into one agent session	spec.md
Build	Handwritten code & retrospective docs	AI-generated code guided by versioned institutional knowledge	plan.md & CLAUDE.md
Test	QA gates at stage boundaries	Continuous evals woven through implementation	diff
Deploy	Humans review every line	Agentic review layers; Governance via deterministic hooks	review.md
Maintain	Humans watch prod for bugs	Agents monitor live deployments and automatically write fixes	intent.md (new)

The Source of Truth Matrix

Repo as Truth



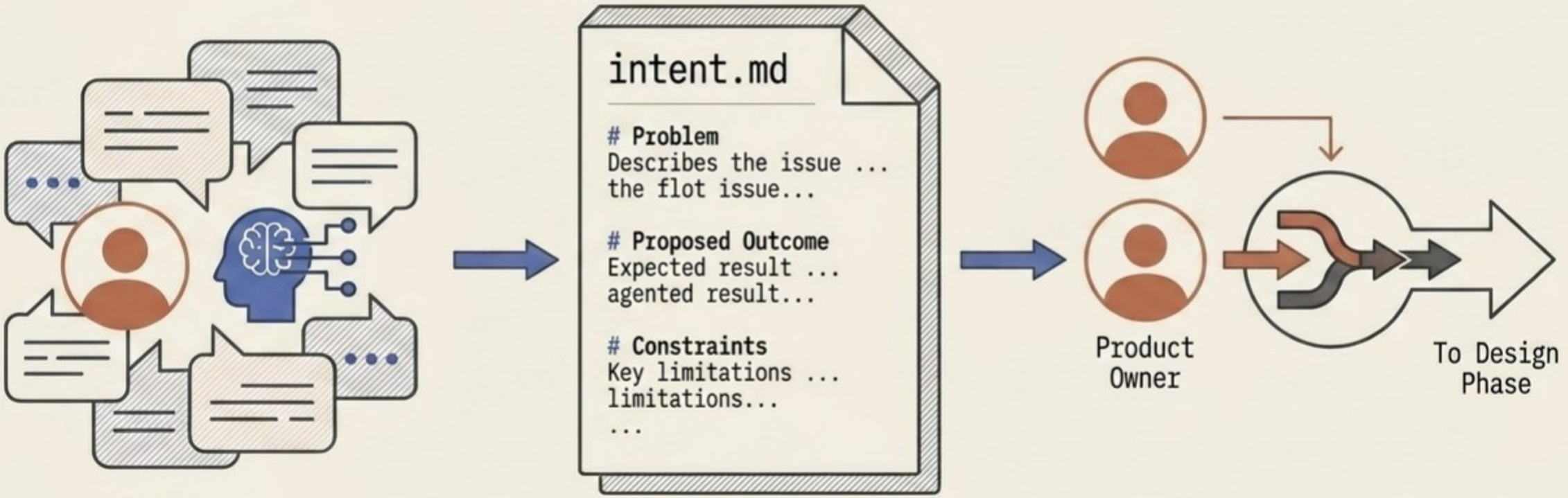
Legacy System as Truth



Both models work. Linkage is the minimum bar. Every markdown artifact must note the **record ID**, and every legacy record must contain the **commit SHA**.

Play 1: Capture Intent at the Source

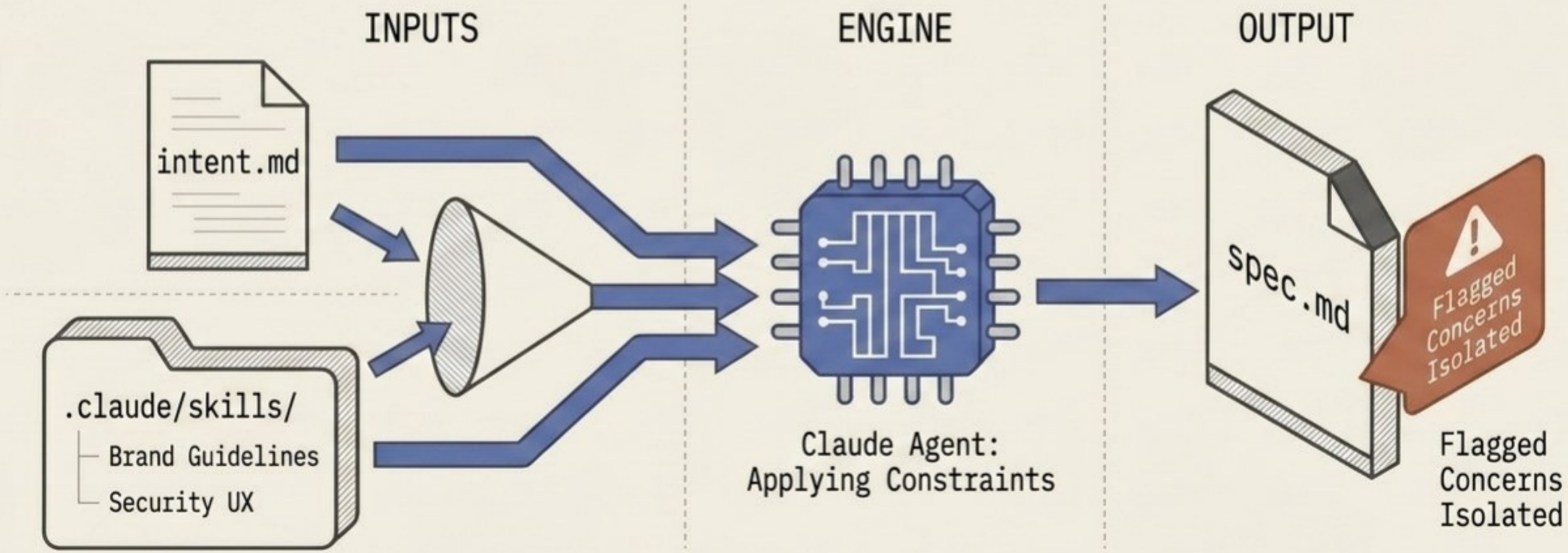
Ideas no longer wait for refinement meetings. The originator brainstorms directly with Claude to produce a proto-spec in their own words.



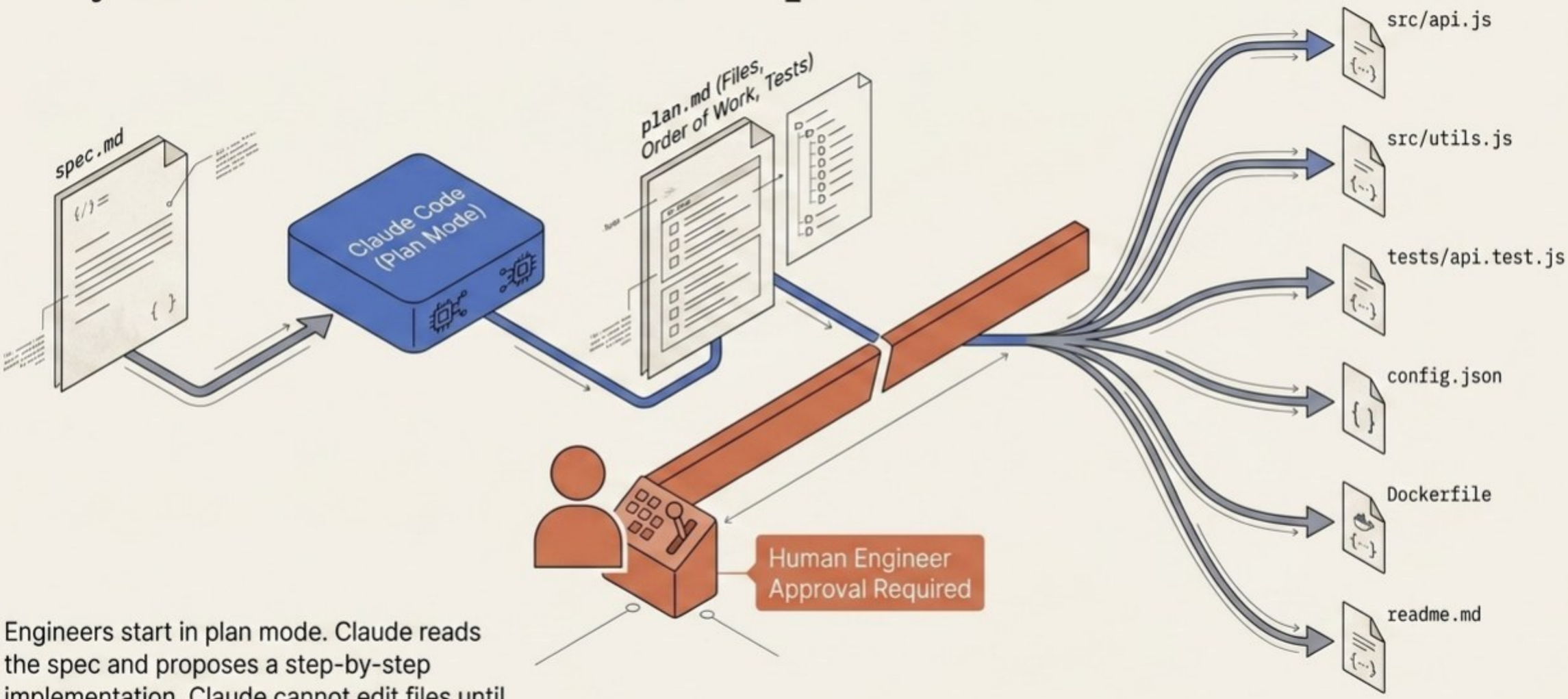
1. Brainstorming Session
2. Claude Generates Proto-Spec
3. Human Review & Merge

Play 2: Compressing Requirements & Design

Instead of weeks of analyst translation, the product owner runs a single session. Claude reads the intent, applies organizational skills, and outputs an engineering-ready spec with policy violations flagged immediately.



Play 3: Never code without a plan.



Engineers start in plan mode. Claude reads the spec and proposes a step-by-step implementation. Claude cannot edit files until the engineer explicitly accepts `plan.md`.

Internal Governance as Code

Control shifts from human habits to version-controlled files inside the repository, moving from advisory guidelines at the surface to deterministic blocking at the bedrock.

Layer 1: CLAUDE.md

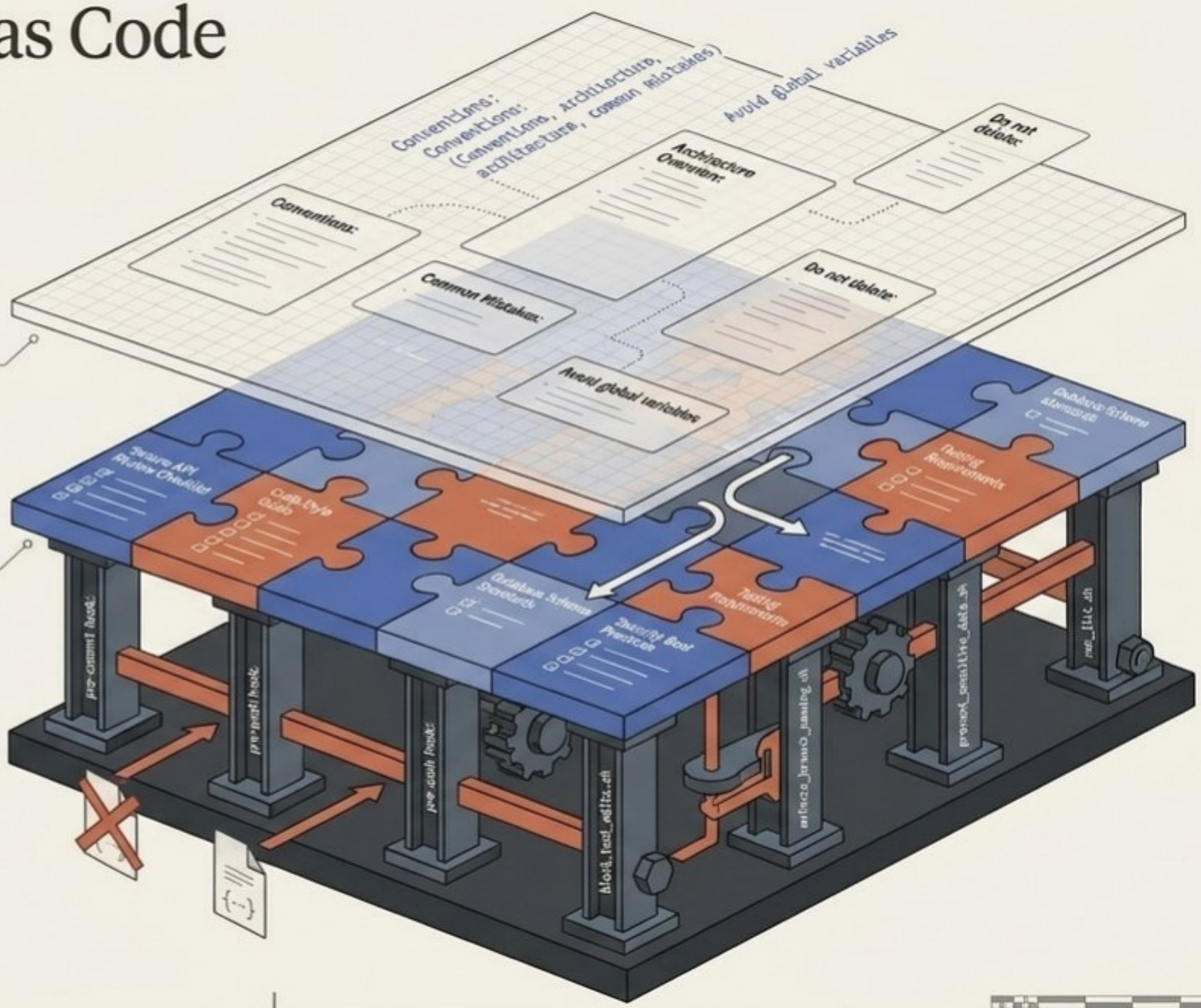
Context for the agent (Conventions, architecture, common mistakes)

Layer 2: Skills

Encoded institutional knowledge (e.g., Secure API Review checklist)

Layer 3: Hooks

Deterministic shell scripts (e.g., blocking edits to test files)

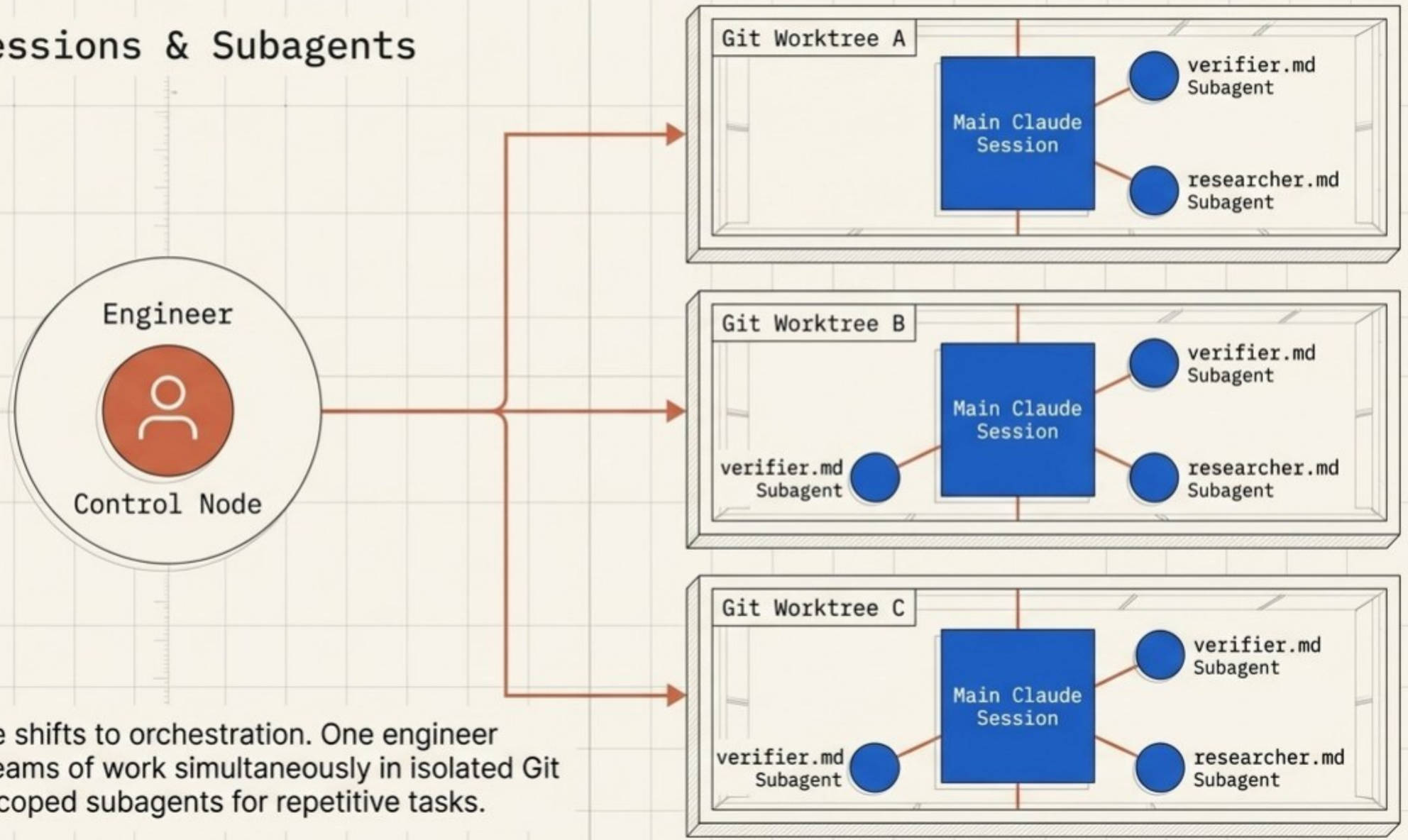


The Governance Matrix

	Skills	Hooks	Evals
Type of Control	Advisory (Claude is likely to apply it)	Deterministic (Blocks or allows the action)	QA Gate (Checks aggregate output)
Execution Timing	During code creation	Milliseconds before an action executes	Post-commit in the CI/CD pipeline
Best Used For	Brand guidelines, API naming conventions	Preventing edits to protected paths, credential blocking	Regression testing for agent behavior

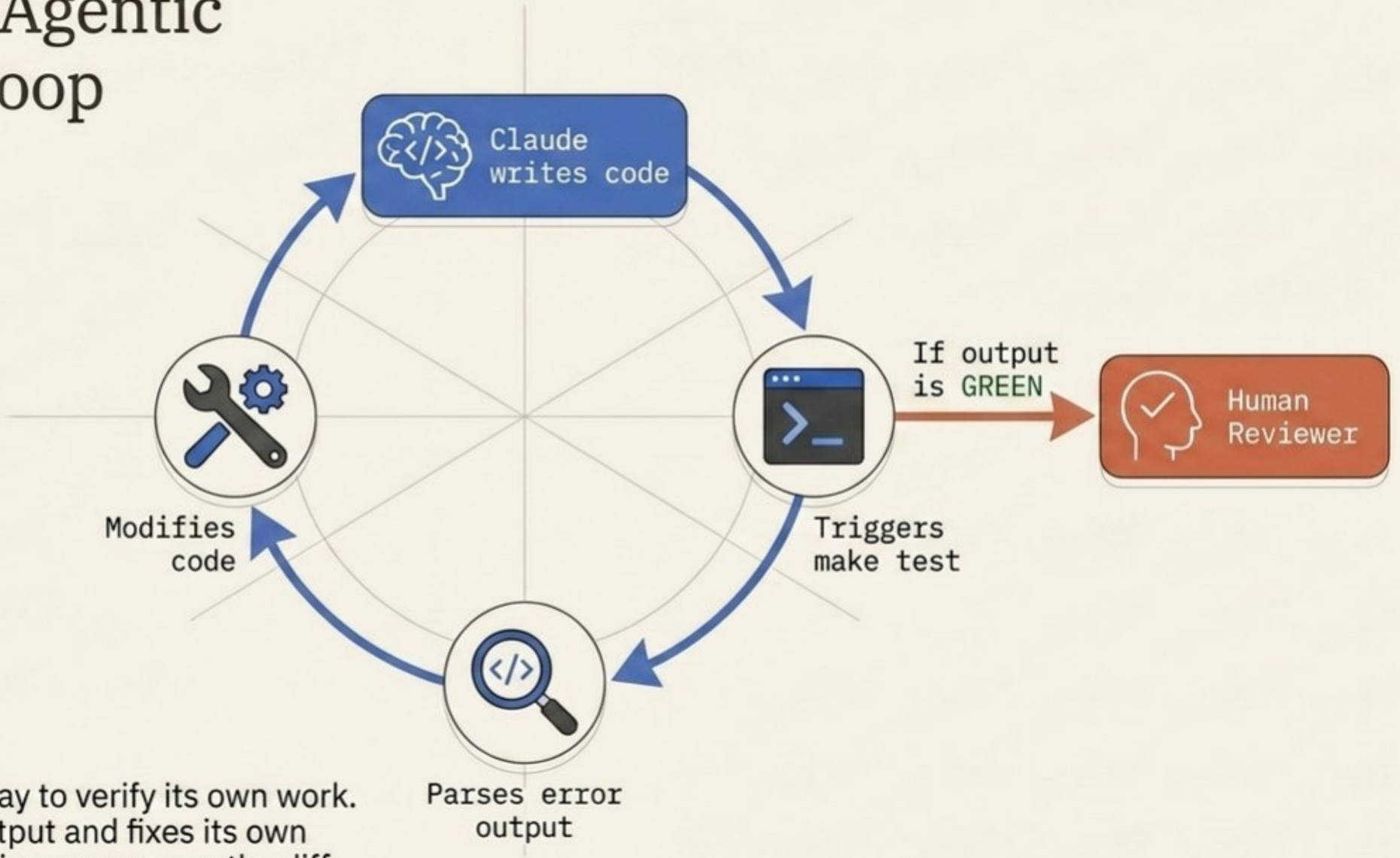
The Orchestrator Model

Parallel Sessions & Subagents



The engineer's role shifts to orchestration. One engineer drives multiple streams of work simultaneously in isolated Git worktrees, using scoped subagents for repetitive tasks.

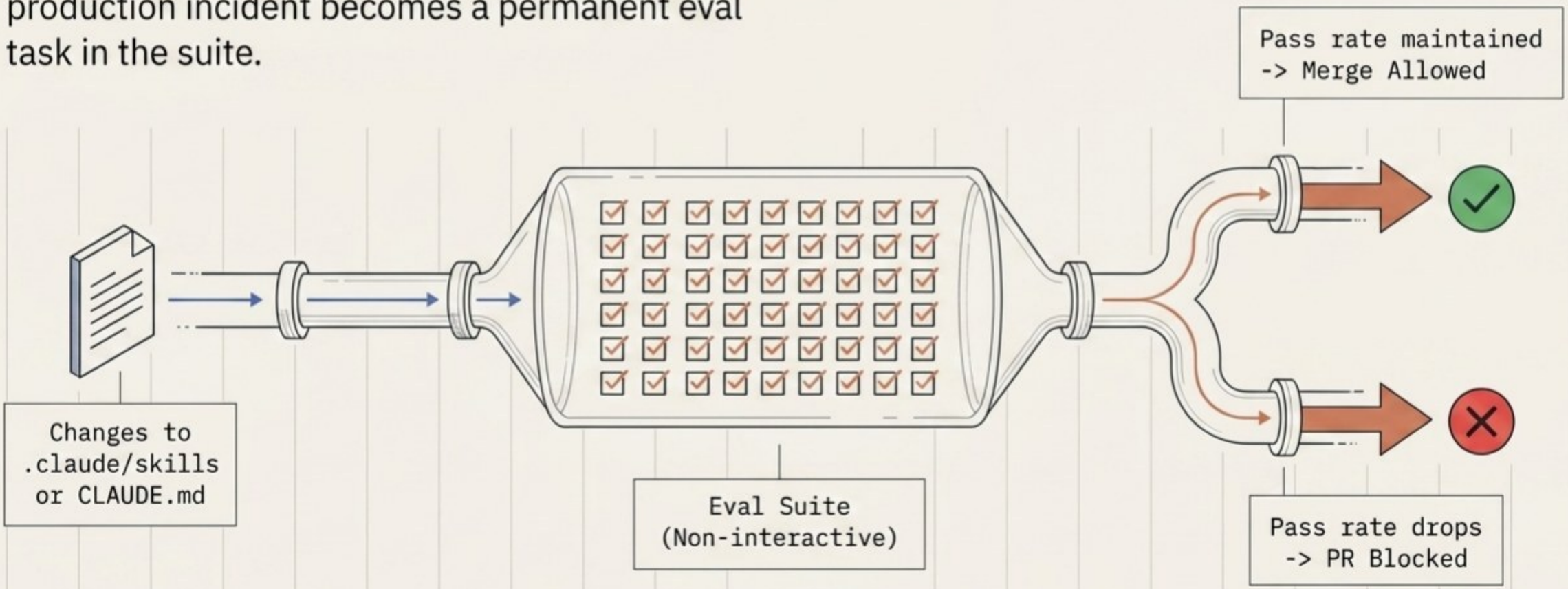
Play 4: The Agentic Feedback Loop



Always give Claude a way to verify its own work. A session checks its output and fixes its own mistakes before an engineer ever sees the diff.

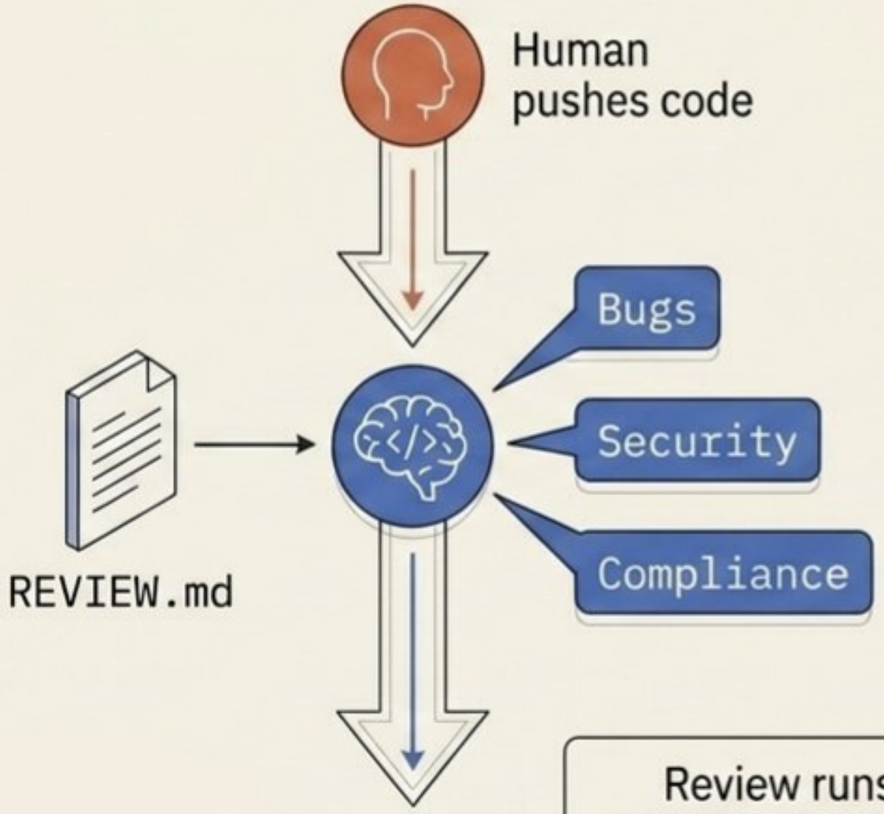
Continuous Evals in CI

The configuration that steers the agent must be regression-tested like the code it writes. Every production incident becomes a permanent eval task in the suite.

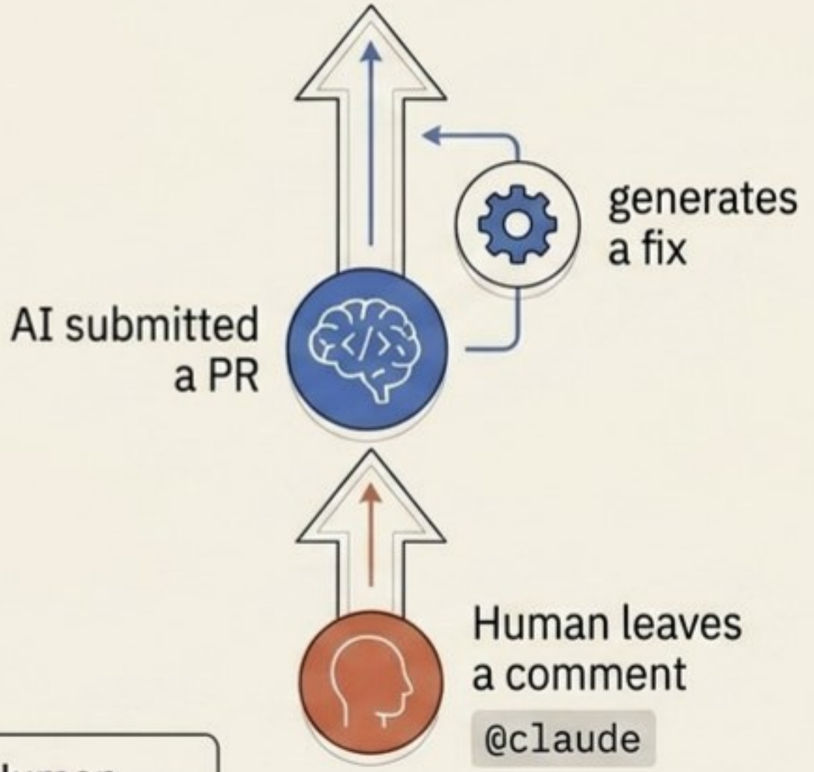


Play 5: AI in the PR Loop

Incoming PRs



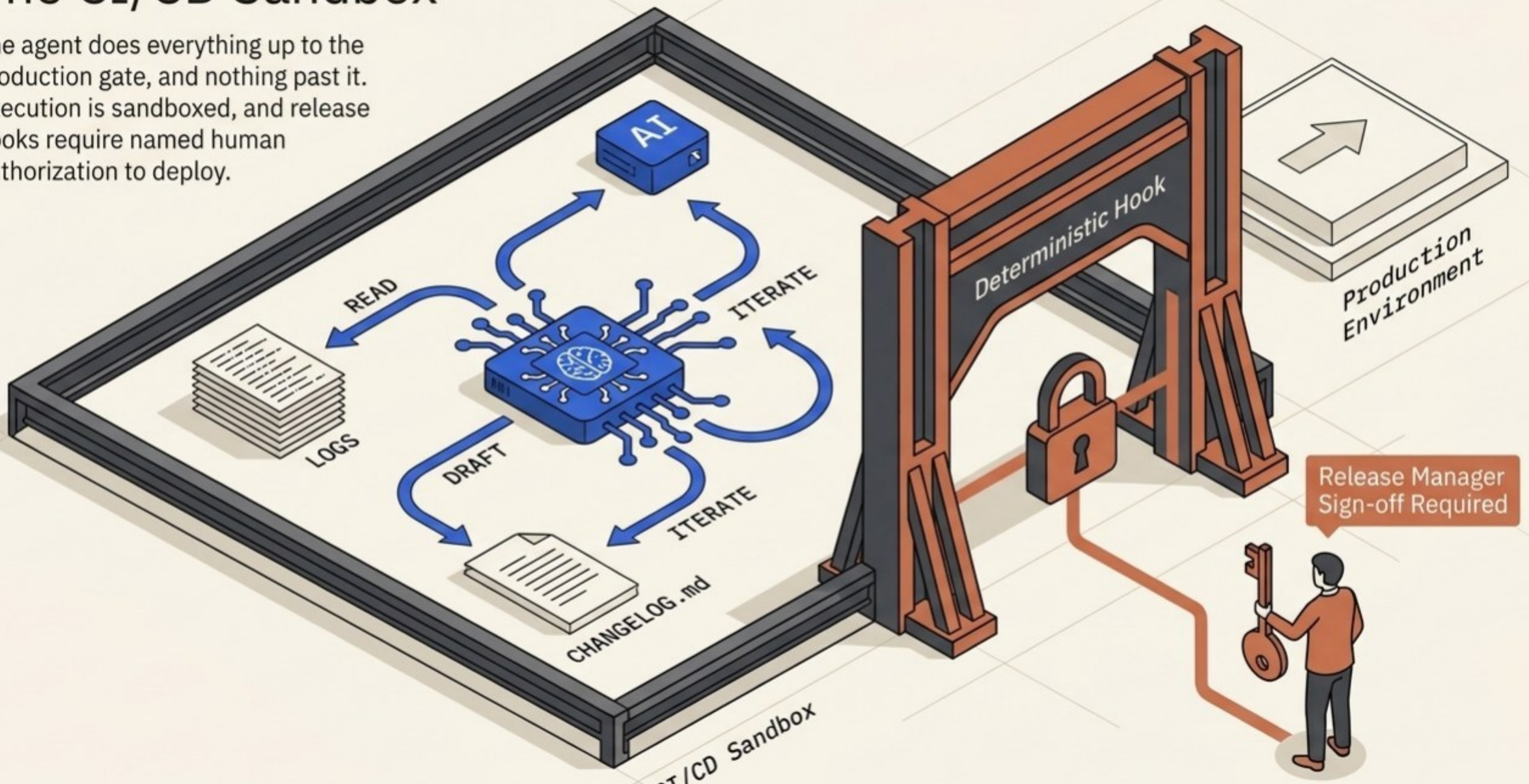
Outgoing PRs



Review runs in both directions. Human attention moves up a level: judging intent and risk, rather than hunting for syntax errors.

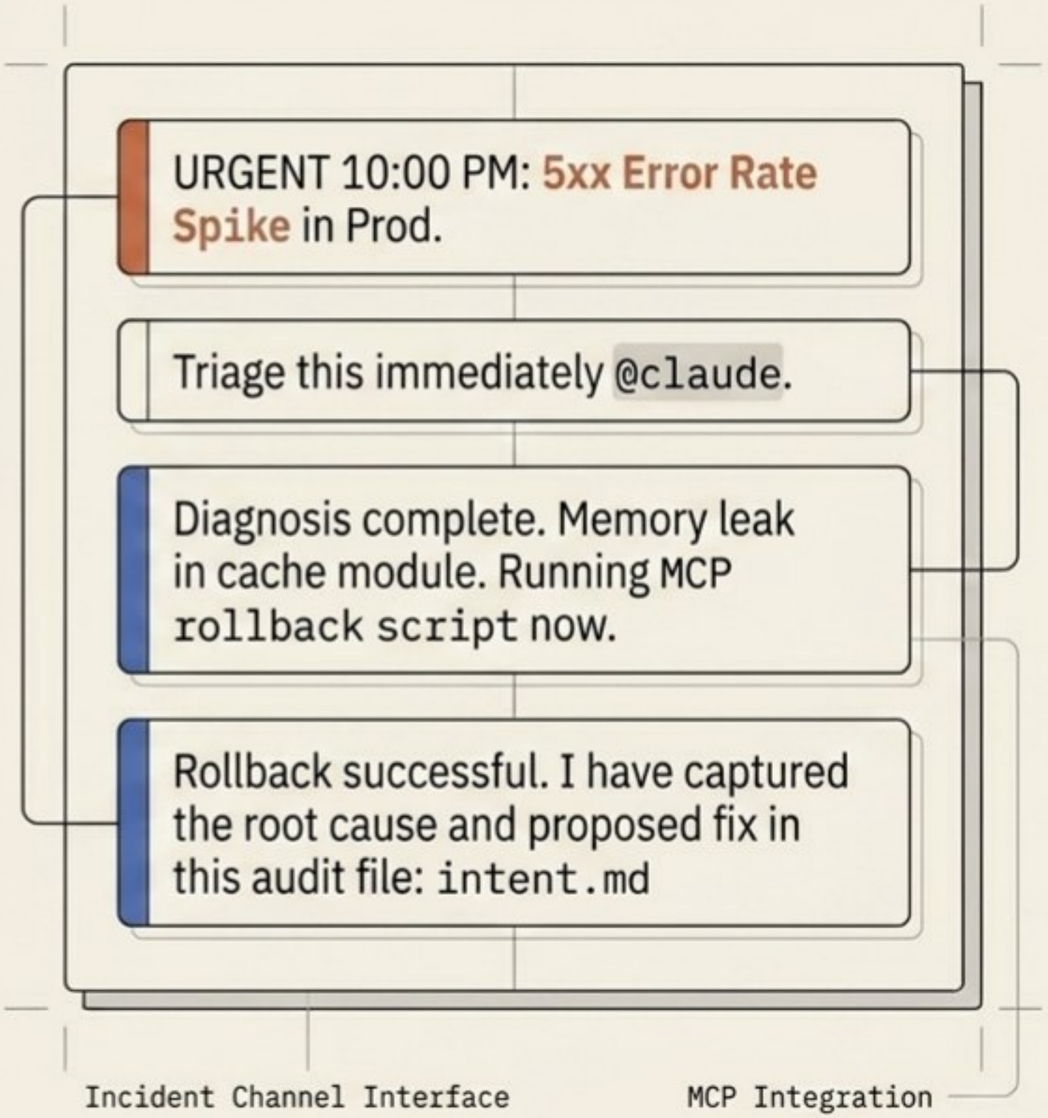
The CI/CD Sandbox

The agent does everything up to the production gate, and nothing past it. Execution is sandboxed, and release hooks require named human authorization to deploy.



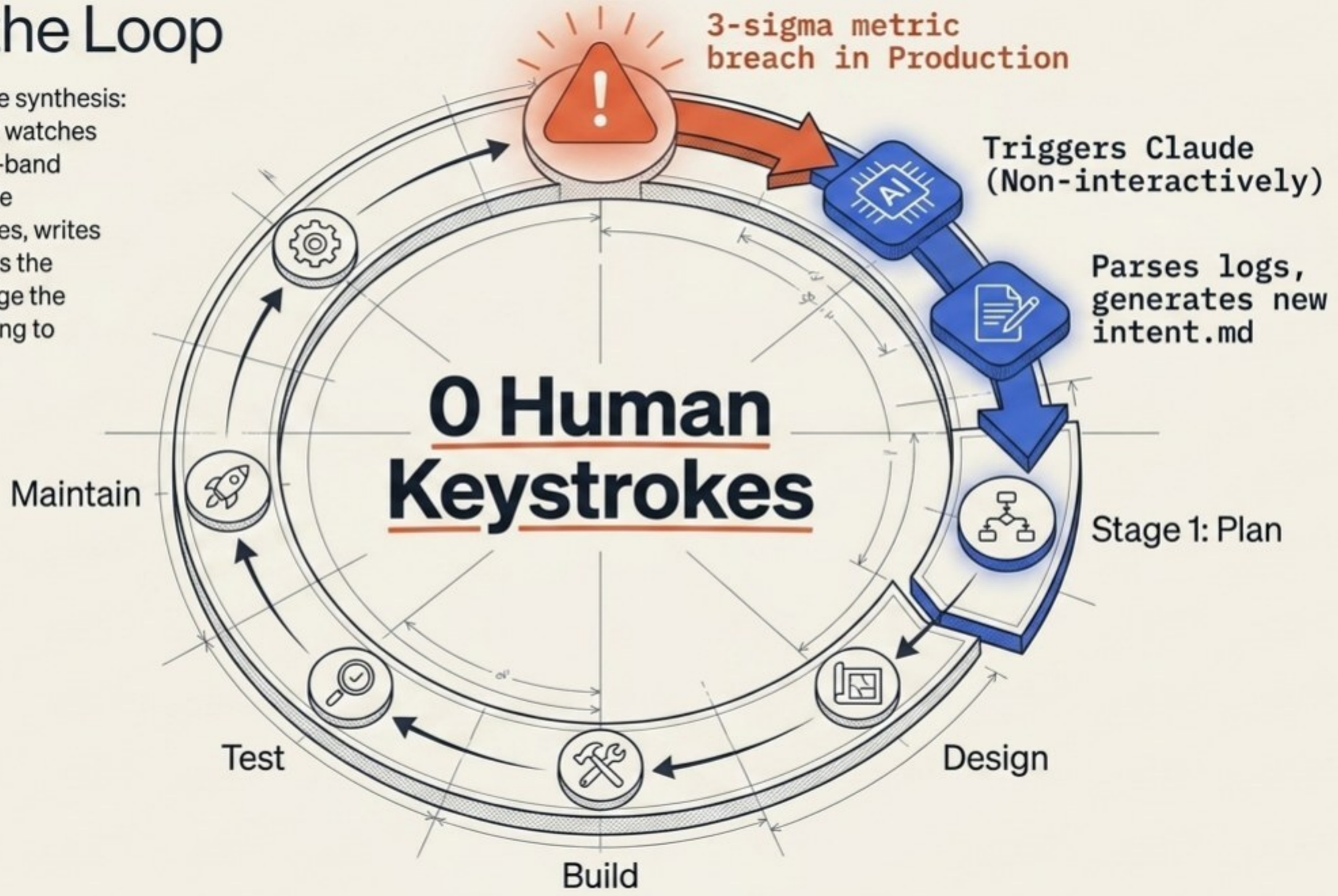
Play 6: Claude on Call

Using Claude Tag, the agent acts as a first responder in incident channels. It triages alerts, explores options via MCP, and captures the institutional knowledge directly into the audit trail.



Closing the Loop

The ultimate AI-Native synthesis: A deterministic script watches production. A control-band breach invokes Claude statelessly. It diagnoses, writes the intent, and restarts the pipeline. Humans triage the queue, no longer having to start it.



The loop keeps running. Human judgment stays above it.

The AI-Native SDLC doesn't remove humans—it elevates them from manual bottlenecks to orchestrators of a continuously running system.