Make Coding Agents Verifiable Before You Make Them Autonomous

Coding Agents Alpha Tracker

2026-07-17

Make Coding Agents Verifiable Before You Make Them Autonomous

By Coding Agents Alpha Tracker • July 17, 2026

Today's practical lesson is how to design a control plane around coding agents: centralize signals, validate outcomes, codify orchestration, and preserve human judgment at the highest-leverage points. Also: managed-agent updates from Gemini, a Deep Agents Code setup, and a multi-model Devin workflow.

TOP SIGNAL

The agent control plane is where the real work is moving. Factory's Eno Reyes says humans intervene most heavily on whether a proposed change and its plan/spec are actually the right ones; agents can then handle QA, code review, and security analysis. [1] Armin Ronacher's counterweight is blunt: remove every bit of friction and teams risk output they cannot understand or maintain—so keep review and operational checks while automating the toil around them. [2]

TRY THIS

- **Build a signal-to-plan inbox before you build an auto-merge loop.** Instrument customer Slack, GitHub issues, internal product discussions, telemetry, and other feedback into one pipeline. Have agents triage against a concise product-prioritization document, then make a human explicitly approve or reshape the resulting plan before agents implement, test, review, and analyze security. Factory keeps this guidance under roughly 2,000 lines. [1]
- **Turn a large task into a "mission" with validators.** Before implementation, force the agent to enumerate concrete outcomes and attach a

validator to each: deterministic checks where possible, an LLM-based verifier only where necessary. Keep the loop running until every validation passes; run an agent-readiness check first. This is Factory's approach for decomposing large engineering jobs. [1]

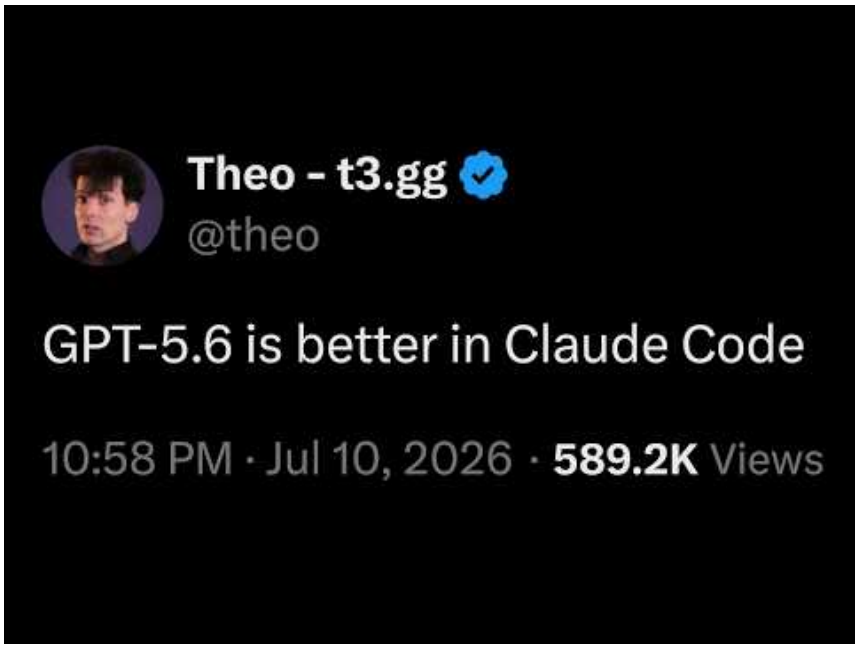
- **Make orchestration code, not emergent behavior.** Theo's Claude Code workflow has the model write a JavaScript workflow file up front—stages, prompts, and sub-agents—then execute it top-to-bottom. Put the workflow convention in your global `agents.md` or system prompt so it becomes reusable; Theo reports these bounded workflows end cleanly and used about one-quarter of the tokens for comparable tasks in his testing. [3]
- **Treat permissions as a workflow decision.** Keep autonomous agents to local, reversible actions such as edits and tests; require confirmation for destructive, hard-to-reverse, shared, or externally visible actions. This is not theoretical: reported Codex file deletions most commonly involved full-access mode without sandboxing or auto-review, followed by a mistaken `$HOME` deletion during temporary-directory setup. [3, 4]

WHAT SHIPPED

- **Gemini API managed agents:** Google added cost controls, a free tier, and initial triggers for scheduled agent tasks. Related announcement. [5]
- **Deep Agents Code + NVIDIA Nemotron 3 Ultra:** LangChain highlighted a terminal-agent setup via Baseten with skills, sub-agents, MCP support, and LangSmith tracing; the cited model configuration is 550B parameters and up to 300 tokens/sec. [6]
- **Devin's current multi-agent workflow:** Riley Brown's hands-on walkthrough shows its Sessions/Kanban view for concurrent agents and per-agent model selection across SWE 1.7, Claude Fable 5, GPT-5.6 Luna, and GLM 5.2 High. The practical takeaway is less the model menu than the workflow: queue work, preview in-app, iterate with screenshots, and deploy through connected GitHub/Vercel tooling. [7]
- **Model routing is becoming a product control, not a developer habit.** Factory's routing exposes an admin-set quality/cost slider and learns from a company's task distribution; the company says typical savings are 30–50%, with the explicit tradeoff being quality. [1]

GO DEEPER

- **4:40–5:44 — Theo on bounded, code-defined subagent workflows.** A concise explanation of why a workflow file can be easier to control—and terminate—than agents spawning subagents opportunistically. [3]



I need you to hear me out (it's REALLY good) (4:39)

- **58:50–60:47 — Factory's "missions" model.** Watch the outcome/validator framing: define what success means, select deterministic or LLM validation, and do not let the agent declare completion prematurely.

[1]

The best AI agents need more humans than you think (57:53)

- **26:39–29:25 — Armin Ronacher on guardrails that preserve human judgment.** Concrete examples include linting architectural constraints, automatically fixing mechanical issues, and escalating migrations or dependency changes for human review. [2]

A Year of Agents | Armin Ronacher | CodeCrafts 2026 (26:39)

Editorial take: the winning agent setup is not maximum autonomy—it is a system that makes plans reviewable, outcomes verifiable, and risky actions hard to perform by accident.

Sources

1. The best AI agents need more humans than you think
2. A Year of Agents | Armin Ronacher | CodeCrafts 2026
3. I need you to hear me out (it's REALLY good)
4. Quoting Thibault Sottiaux
5. X post by @OfficialLoganK
6. X post by @LangChain
7. Devin AI: The Full Beginner's Guide (Better Than Claude Code?)