Deep Agents Deploys, Advisor Routing Lands, and Harness Design Takes the Lead

Coding Agents Alpha Tracker

2026-04-10

Deep Agents Deploys, Advisor Routing Lands, and Harness Design Takes the Lead

By Coding Agents Alpha Tracker • April 10, 2026

The strongest practical theme today: coding-agent performance is increasingly a harness problem, not just a model problem. This brief covers LangChain's Deep Agents deploy launch, Anthropic's advisor routing pattern, Claude Code and Cursor upgrades, and concrete workflow patterns from engineers running agents in production.

TOP SIGNAL

Today's best signal is a convergence: Ryan Lopopolo (OpenAI), Vincent Potch (OpenClaw), and Izzy Miller (Hex) are all describing the same shift — coding-agent gains are now coming from **harness design** more than prompt cleverness. The pattern repeats across talks: encode quality as docs/lints/reviewer agents, shrink or search tool surfaces, and run agents in parallel lanes with real evals over long horizons [1, 2].

TOOLS & MODELS

- **LangChain: Deep Agents deploy (beta).** New single-command deploy path for a model-agnostic, open-source agent harness. You configure `AGENTS.md`, `skills`, `mcp.json`, choose a model + sandbox, and it spins up a LangSmith deployment with MCP, A2A, Agent Protocol, human-in-the-loop, and memory endpoints [3]. LangChain is explicitly positioning it as an open alternative to Claude Managed Agents, with memory ownership as the key differentiation [3].
- **Anthropic: advisor/executor routing.** Claude Platform is adding an "advisor" strategy: pair **Opus** as the advisor with **Sonnet** or **Haiku** as the executor for near-Opus-level intelligence at lower cost [4]. Anthropic's

eval claim: **Sonnet + Opus advisor** scored **+2.7 points** on SWE-bench Multilingual versus Sonnet alone while costing **11.9% less per task** [5, 6].

- **Claude Code: real product upgrades, not just model chatter.** Anthropic added a `fileSuggestion` setting so large-codebase users can plug in custom indexes like Sourcegraph or internal systems [7]. Boris Cherny says the latest improvement shipped in **Claude Code v2.1.85**, where a Claude-driven port from Rust+NAPI to native TypeScript made `@`-mentions **3x faster at P99** [8, 9, 10]. Separately, Claude Code now has a setup wizard for **Amazon Bedrock** and **Google Vertex**, plus detection for pinned older models with suggestions for newer ones [11, 12].
- **Cursor: cloud-agent ergonomics keep getting better.** Cursor cloud agents can now attach **demos and screenshots** to the PRs they open, so teammates can review artifacts directly in GitHub [13]. Jediah Katz also showed a smaller but important feature: agents can now **wait on background jobs** and wake back up based on log output [14].
- **TanStack AI Code Mode is a notable execution-layer bet.** The idea: let the model write and execute **TypeScript** instead of chaining tools, because LLMs are strong at TS but weak at math/orchestration [15]. Claimed benefits are **1 call instead of N, parallel execution, fewer tokens, and correct results** [15].
- **OpenAI: Codex capacity is becoming a product tier.** OpenAI launched a **$100/month Pro tier** with **5x more Codex usage** than Plus, targeted at longer, high-effort sessions, plus a limited-time promo of up to **10x Plus usage** through May 31 [16]. Separate signal: Codex is now at **3 million weekly users**, up from **2 million** less than a month earlier [17].

WORKFLOWS & TRICKS

- **Use the agent as a performance engineer, not just a code generator.** Boris Cherny's prompt pattern is excellent: tell Claude to port an implementation, require it to pass the **original** test suite, benchmark against the old path, and keep iterating until it proves it is faster [18]. He then tightened the loop with explicit profiling goals ("hoping for p99 $<$ 10ms") and follow-up refinement prompts, which led to concrete wins like pre-computing without blocking the main thread and avoiding NAPI overhead for small result sets [18, 19].
- **Turn repo standards into machine-enforced prompts.** Ryan Lopopolo's playbook: document non-functional requirements once, then reinforce them everywhere — `agents.md`, reviewer agents, lints, tests, CI comments, and error messages with remediation steps [1]. His concrete examples: enforce retries/timeouts on network code, add tests that cap

file length, and run security/reliability review agents continuously so the model keeps getting reminded what "good" looks like [1].

- **Prefer deep modules with simple interfaces.** In the AIE Europe talk, shallow-module sprawl was called out as actively hostile to AI navigation: too many tiny blobs, too many dependencies, too much searching [1]. The better pattern is fewer, deeper modules with well-designed boundaries; then test at the interface and let the agent work more freely inside the boundary [1].
- **Run agents in "swim lanes."** Vincent Potch's setup is a useful mental model: keep separate lanes for stable refactors/CI, feature work, and P0/P1 monitoring, and only babysit the risky lanes [1]. Stable refactors can often run with minimal supervision; feature and incident lanes need a tighter conversation loop [1].
- **Shrink tool surfaces before you add more tools.** Hex ended up with roughly **100k tokens worth of tools**, which Izzy Miller flatly called "too many" [2]. Their response is timeless: consolidate families of similar tools, add **tool search/tool retrieval**, and use specific tools when you want behavioral guidance instead of dropping the model into fully generic code execution [2].
- **Evaluate compounding behavior, not just day-zero accuracy.** Hex's most interesting eval pattern is a mix of small handcrafted "trap" sets and a **90-day simulation** where the agent answers tickets, updates knowledge, and keeps operating as the environment changes [2]. In that setup, Sonnet reportedly went from about **4% on day 0** to **24% on day 90** [2].

PEOPLE TO WATCH

- **Ryan Lopopolo** — High signal because he is specific about the boring stuff that actually moves quality: QA plans, reviewer agents, lint/test prompts, and durable repo-level guidance [1].
- **Boris Cherny** — Worth following because he is using Claude Code on enterprise-scale codebases and showing exact prompts, perf targets, and benchmark loops instead of vague "it feels faster" claims [8, 18, 9].
- **Izzy Miller** — One of the best current sources on context engineering for agents: tool explosion, long-running orchestration, scratchpad queries, memory/guide conflicts, and why data agents are harder to verify than coding agents [2].
- **Vincent Potch** — Useful for anyone moving from one-agent demos to actual multi-agent operations. His "factory manager" framing is one of the cleanest models I've seen for running many parallel sessions without pretending tokens are the bottleneck [1].

- **Andrej Karpathy** — His thread matters because it explains why AI discourse keeps splitting in two: casual users see weak free-tier behavior, while engineers on frontier coding models see systems that can restructure codebases and find vulnerabilities because technical domains have verifiable rewards and strong B2B incentives [20].

WATCH & LISTEN

- **AIE Europe — Vincent Potch on "swim lanes" (3:04:43-3:05:58).** Best quick explanation of how to supervise 5-20 coding agents at once: stable refactors can run mostly unattended, while feature and incident lanes need active conversation and triage [1].

AIE Europe Day 1: Keynotes & OpenClaw/Personal Agents ft Google Deepmind, OpenAI, Vercel, & more (184:43)

- **AIE Europe — deep modules > shallow modules (8:42:09-8:44:29).** A crisp case for reorganizing AI-heavy codebases around deep modules with simple interfaces so agents can navigate, modify, and test them more reliably [1].

AIE Europe Day 1: Keynotes & OpenClaw/Personal Agents ft Google Deepmind, OpenAI, Vercel, & more (522:09)

- **Hex — evaluate the flywheel, not just day zero (1:00:20-1:04:21).** Izzy Miller walks through a 90-day simulation with changing warehouse state, inbound tickets, and proactive agent work. This is one of the best arguments for long-horizon evals I've seen [2].

How Hex Builds AI Agents: Making Agents Reason Like Human Data Analysts | Izzy Miller, AI Engineer (60:20)

PROJECTS & REPOS

- **OpenClaw.** Peter Steinberger says the project is only **five months old** but already has about **30k commits**, is closing in on **2,000 contributors**, and is approaching **30,000 PRs** [1]. The architecture is also shifting toward a plugin model so teams can swap in their own memory/wiki/dreaming components without forking the whole thing [1].
- **Deep Agents / Deep Agents deploy.** The important part is not just the beta launch — it's that the underlying harness is **MIT-licensed**, available in Python and TypeScript, built around open standards like `AGENTS.md`, Agent Skills, MCP/A2A/Agent Protocol, and can be self-hosted so memory stays yours [3].
- **TanStack AI Code Mode.** Notable framework pattern to watch: move orchestration into executable TypeScript instead of ever-growing tool chains. The pitch is fewer calls, fewer tokens, and better parallelism for complex app logic [15].

Editorial take: the edge is moving from "which model is best?" to "how well did you design the repo, tool surface, eval loop, and operator workflow around the model?" [1, 2]

Sources

1. AIE Europe Day 1: Keynotes & OpenClaw/Personal Agents ft Google Deepmind, OpenAI, Vercel, & more
2. How Hex Builds AI Agents: Making Agents Reason Like Human Data Analysts | Izzy Miller, AI Engineer
3. Deep Agents Deploy: an open alternative to Claude Managed Agents
4. X post by @claudeai
5. X post by @alexalbert___
6. X post by @claudeai
7. X post by @bcherny
8. X post by @bcherny
9. X post by @bcherny
10. X post by @bcherny
11. X post by @morganlunt
12. X post by @_catwu
13. X post by @cursor_ai
14. X post by @jediahkatz
15. X post by @tan_stack
16. X post by @OpenAI
17. X post by @thsottiaux
18. X post by @bcherny
19. X post by @bcherny
20. X post by @karpathy