

Kimi K3.1 Pro nears frontier scores at half Opus pricing, and Cursor ships computer use for agent self-verification

Coding Agents Alpha Tracker

2026-10-24

Kimi K3.1 Pro nears frontier scores at half Opus pricing, and Cursor ships computer use for agent self-verification

By Coding Agents Alpha Tracker • October 24, 2026

Moonshot's Kimi K3.1 Pro posts near-frontier scores ahead of an open-weights release. Separately, Cursor, Warp and Addy Osmani are making “the agent shows proof it works” a standard step. Also: Codex CLI updates, lessons from OpenAI's Codex team, Theo's multi-model merge gate, and LangChain's retrieval patterns.

Kimi K3.1 Pro: near the top, cheaper, open weights coming

Moonshot's Kimi K3.1 Pro is live on the Kimi API, and Moonshot says “Full model weights will be released by Nov 4, 2026.” It costs \$2/M input and \$10/M output tokens, which Simon Willison notes is the same as GPT-6.1 Sol and half the price of Claude Opus 5.5. A Highspeed version is 2–3× faster and costs twice as much, except cached input tokens, which cost the same [1]. Artificial Analysis scored it 55. That puts it behind Opus 5.5 (58) and ahead of Claude Fable 5.1, GPT-6 Astra and Gemini 4 Argon, all at 53. Willison calls it the first time a soon-to-be open-weights model “has scored this highly against the frontier” [1]. Moonshot says it has a 1M-token context and 2.8T total / 104B active parameters, and can do full-project coding, including games in Unreal, Unity and Godot. It is also available in Kimi Code [2].

You can test it from any OpenAI-compatible client. Willison used `llm openai endpoint https://api.moonshot.ai/v1 --key kimi -m kimi-k3.1-pro "..."` `--responses -o reasoning_effort low`, sending stdout and stderr to separate files. That command doesn't log results, so this keeps both the answer

and the reasoning trace. The model supports low, medium and max reasoning effort [1].

Agents that bring back proof they finished

Cursor agents can now control your computer. They can click and type in desktop apps such as the iOS Simulator to test the features they build, not just in a browser [3]. Cursor turns it on by itself when a task needs it, or you can @-mention Computer Use in your prompt. Update Cursor to get it [4].

Warp has the same idea in Warp Factories. Each agent gets a computer to use the app it built, then posts screenshots and recordings for a person to review. About 3–4 minutes of activity gets cut down to 15 stitched clips. Warp says this “dramatically” cut time to merge, especially for projects that are hard to mock locally [5]. **Addy Osmani** does something similar. On top of E2E tests, he has Claude return recorded videos and screenshots as evidence a task is “done.” He bets verification will be one of the most important problems this year [6].

There was less agreement on how much humans should check tests. Kent C. Dodds says he hasn’t written a test all year, and “the tests were the first thing to go for me” [7, 8]. DHH says bug reports written by agents are “excellent on average,” while human-written ones are terrible [9].

Codex: CLI updates and the team’s lessons

Tibo of OpenAI listed new Codex CLI features: reusable prompt presets, inline model and permission settings, and enterprise MCP sign-in. It also fixes failures when resuming long sessions and adds Windows corporate-proxy authentication. He says CLI usage has doubled since DevDay [10]. Changing the reasoning setting partway through a conversation no longer invalidates the cache [11]. Riley Brown points out a catch now that Codex lives inside the ChatGPT desktop app: skills you made in Codex work in local GPT Work sessions but not in cloud sessions [12].

In a Datadog fireside chat, **Alexander Embiricos** said Codex Cloud was “too early.” Codex took off after the team switched to a CLI, an IDE extension and a local app [13]. His takeaway is to start with quick, useful work in the user’s local context, then hand over more over time. Agents need access to your tools and a way to reproduce problems and measure results [13]. He recommends reading OpenAI’s open-source harness, and suggests companies choose a harness for automation using private evals built from their own tasks [13].

He also described a cost of cheap code. OpenAI’s open-source Rust harness now accepts external PRs only for issues the team has already approved. Internally, PRs need an RFC with architectural agreement first, and agent skills help people write those RFCs. A small group of tech leads gates the code [13].

Using several models together

Theo wrote a “council” skill. Before his agent is allowed to merge, it has to argue its case with Opus 5.5, Muse Wonder Full and GPT-6.1 Sol [14]. He uses Muse for orchestration and for rewriting docs, skills and MCP descriptions, and says it is good at steering other models [15]. It also follows style requests like “less words and simple sentences,” where Claude takes “a LOT of effort” to change [16]. For remote work, running `npx t3 connect` gives Claude Code remote support in T3 Code. Theo says fewer than 5% of his prompts run on his laptop, and that remote access has more than 10×’d how hard he runs agents [17, 18].

Retrieval patterns that keep context clean

LangChain’s guide to retrieval in Deep Agents suggests these steps: - **Search cheaply first.** Use `glob` for file names and `grep` for exact text, then `read_file`, which returns 100 lines by default with offset and limit options [19]. - **Add semantic search when wording can vary**, and let the agent decide when to search again [19]. - **Retrieve, offload, delegate.** The search tool writes results to files and returns only the paths. Subagents read one file each, in parallel, and return key facts with sources. The main agent combines them [19]. - **Treat retrieved text as possible prompt injection**, and check that answers cite the sources you expect [19].

Will Brown of Prime Intellect makes a related point: most developers can’t name their coding agents’ pass rate. When a terminal experiment proves useful, turn it into a skill or cron job, then add evals and versioning [20].

Security fallout

OpenAI’s agents, red-teaming its training environment, found an unauthenticated RCE in **Brotli**, the compression library used in Node.js, Bun and Envoy. A client that accepts compressed input, such as `curl --compressed`, can be taken over by a malicious server [21]. Brotli has published new releases and a CVE, and downstream projects are shipping patches [21]. On the OpenCode breach, which affected 1.4% of Console users, OpenCode’s Dax told ThePrimeTime that the attacks are automated and run in parallel. Their defense uses agents running 24/7 that cost “at a minimum several thousand a day” [22].

Sources

1. Introducing Kimi K3.1 Pro
2. X post by @Kimi_Moonshot
3. X post by @cursor_ai
4. X post by @cursor_ai
5. X post by @varoonkodithala

6. X post by @addyosmani
7. X post by @kentcdodds
8. X post by @kentcdodds
9. X post by @dhh
10. X post by @thsottiaux
11. X post by @thsottiaux
12. X post by @rileybrown
13. Fireside Chat: Alexander Embiricos (OpenAI) and Yanbing Li, CPO, Datadog
14. X post by @theo
15. X post by @theo
16. X post by @theo
17. X post by @theo
18. X post by @theo
19. Context Engineering Starts with Retrieval
20. The Boring Version of Continual Learning
21. X post by @OpenAI
22. OpenCode got hacked