

OpenAI speeds up Astra and Sol in every Sign in with ChatGPT tool as Cursor's SDK adds mid-run steering

Coding Agents Alpha Tracker

2026-10-06

OpenAI speeds up Astra and Sol in every Sign in with ChatGPT tool as Cursor's SDK adds mid-run steering

By Coding Agents Alpha Tracker • October 6, 2026

GPT-6 Astra and GPT-6.1 Sol now run faster by default in partner harnesses. Cursor's SDK can now steer agents mid-run. Practitioners also argued over how to give agents ways to check their work and whether custom setups still pay off.

Astra and Sol are faster by default, including in third-party harnesses

OpenAI's Tibo says default speed is now about 50% higher for GPT-6 Astra and GPT-6.1 Sol on subscriptions. This covers OpenAI's own products and partners that use Sign in with ChatGPT, "including OpenCode, Pi, Amp, Devin." You don't need to change anything, and he said it would arrive within two hours [1]. He gives throughput as 50 TPS instead of 30 TPS [2]. This was "Day 1" of the 28-day ship-or-reset pledge from the last brief. It goes at the main complaint about Astra raised there, that it is slow without fast mode. If you dropped Astra for speed, it may be worth re-timing it in your harness.

Cursor SDK: steer running agents and get subagent results back

Cursor shipped a set of SDK changes aimed at people who build their own harnesses: - **Steering:** `run.steer()` adds your message to the agent's next turn. If a subagent is mid-task, it moves to the background and keeps working [3]. - **Background subagents report back.** Their results return to the parent

as a follow-up turn on the same run, so `stream()` and `wait()` keep going until every subagent finishes [4]. - **MCP annotations on custom tools.** Tools can carry hints such as `readOnlyHint` and `destructiveHint`, so the model can tell a lookup from a delete before it calls one [5]. - **Your own system prompt.** You can replace Cursor’s system prompt, and rules, skills and tool schemas still load. It is being enabled account by account [6].

Steering and custom system prompts work with local TypeScript agents. Background subagent results come back locally in both TypeScript and Python. Details are in the changelog [7].

Give the agent a way to check its work, and keep that check fast

Addy Osmani’s list of tests worth investing in [8]: - End-to-end tests that simulate real user flows and serve as ground truth. - Property-based tests that state what must never happen and generate thousands of cases to try to break it. - When replacing a system, old-vs-new comparisons on randomly selected inputs. - Fast, deterministic runs. A slow or flaky loop teaches the agent to retry rather than fix the problem.

ThePrimeagen’s Omarchy QA harness shows how hard the speed part is. An agent crawling the desktop in QEMU (unlock the screen, type the password, confirm the desktop is back) first took about 15 minutes. Changes got the median to about 5, and a custom harness got it to about 2 [9]. He is now trying Clef, which he describes as Cloudflare’s jev and chose for its image capabilities. He is aiming for 30 seconds over login, system menu, lock screen, password, back to desktop and a done check; that is a goal, not a result [9]. He says the harder open question is “What is correct? What do we even test?” [9].

Huntley’s Jiti: grow a running Lisp app by talking to it

Geoffrey Huntley released Jiti (write-up, repo). It is a small kernel where you ask for a capability, the model writes Lisp, and the running application keeps that capability until you ask to remove it [10]. An OpenAI model receives the request, operating instructions, tool definitions and the app’s current state. It can inspect functions, read definitions, propose source or run expressions [10]. Accepted definitions are ordinary Lisp, so calling them needs no further inference [10].

Two design choices carry over to other harnesses: - **Separate tools for changing and using the app.** `develop_form` changes the application and `execute_form` uses what already exists. Both share one evaluator, and the split helps the model tell “change the app” from “use it” [10]. - **Safety checks decide what is kept; goals decide when the work is done.** Accepted changes become durable revisions [10].

You can also ask it to save a composition, such as `shout-backwards` built from

`uppercase-string` and `reverse-string`, as a new function that later requests can find and build on [10]. Huntley calls this “the endgame pattern,” but says he “left out how to do verification with it” [11].

Keep your setup vanilla

@thdxr argues that models improve faster than tinkerers can keep up. In his view, custom workflows mostly solve problems that no longer exist, and someone “naively using vanilla codex” is more likely to be getting the state of the art [12]. Kent C. Dodds agrees: aim for as vanilla a setup as possible [13]. One way to get there, he says, is to give agents tools to use instead of instructions they must follow to the letter [14].

Memory and observability

- **Devin “Dreaming.”** Across sessions, Devin builds a memory graph of how you like to work. Overnight it removes stale records and looks for latent information [15]. Cognition is open-sourcing the memory format: graph relationships, history kept as records change over time, stored in git and Markdown, and usable with any agent [16].
- **LangSmith.** At Interrupt NYC, LangChain said its LLM Gateway integrates with Codex, Claude Code and Cursor. It offers per-team, per-user and per-key spend limits, rate limiting and provider fallbacks. Stateful fallbacks, which skip a failing provider for a set period, are coming soon [17]. A new `smithtune` CLI filters LangSmith traces, converted into a standard “trajectory” format, and sends them to Fireworks or Baseten for fine-tuning [17].

Choosing a language when agents write the code

Guillermo Rauch says Vercel’s Turborepo port from Go to Rust was internally controversial because of human migration costs. He now argues that “What’s ‘best for humans’ is no longer necessarily ‘best for business’” [18]. He doubts Rust is the endpoint either [18]. DHH’s case: Rust used to be painful for web apps, but if agents make it productive and you aren’t reading the verbosity, “the equation has changed” [19]. He finished his Campfire conversions with a JavaScript/Express port [20]. He invites people to point an agent at once-campfire and request other languages, and he will take PRs as-is [21].

Smaller items

- **Reviving an old project:** Simon Willison gave Claude Opus 5.5 one prompt on his dormant pure-Python WASM engine: “Evaluate current state of pwasm - then consider what it would take to get the MicroPython and micro JavaScript experiments ... working under it - and what it would take to speed it up.” After 42 commits with minimal follow-up, it

handles almost all of the WASM spec. He says he “wouldn’t trust this thing at all,” hence the alpha tag [22]. His **llm-anthropic 0.30** adds **llm anthropic refresh**, which pulls the model list from Anthropic’s API, and **llm anthropic count**, which counts tokens before you send a prompt [22].

- **Cowork moves to the cloud:** Anthropic now runs both inference and the VM in the cloud, with a sandbox per session. The desktop app handles access to local files, so work continues when your laptop is closed and you can use Cowork from a phone [23].
- **T3 Code** added in-thread visualizations that agents build, using your theme’s CSS variables [24, 25].
- **Beam:** Reflection AI announced an open agentic model with 501B total and 23B active parameters, aimed at coding and agentic tasks. Full weights are due this month [26].
- **Small fixes:** Theo had Opus 5.5 write a working Chrome extension in 1.5 minutes to stop GitHub screenshots opening in a new tab [27].

Sources

1. X post by @thsottiaux
2. X post by @thsottiaux
3. X post by @cursor_ai
4. X post by @cursor_ai
5. X post by @cursor_ai
6. X post by @cursor_ai
7. X post by @cursor_ai
8. X post by @addyosmani
9. X post by @ThePrimeagen
10. an application in lisp you grow by talking to it
11. X post by @GeoffreyHuntley
12. X post by @thdxr
13. X post by @kentedodds
14. X post by @kentedodds
15. X post by @cognition
16. X post by @walden_yan
17. Interrupt NYC: Opening Keynote
18. X post by @rauchg
19. X post by @dhh
20. X post by @dhh
21. X post by @dhh
22. OpenAI DevDay 2026 live blog
23. Quoting Felix Rieseberg
24. X post by @theo
25. X post by @theo

26. X post by @reflection_ai
27. X post by @theo