

GPT-6.1 Sol arrives as a cheap reviewer for Opus 5.5 code, and Codex Cloud 2.0 ships

Coding Agents Alpha Tracker

2026-09-30

GPT-6.1 Sol arrives as a cheap reviewer for Opus 5.5 code, and Codex Cloud 2.0 ships

By Coding Agents Alpha Tracker • September 30, 2026

At OpenAI DevDay, GPT-6.1 Sol launched with very cheap cache reads. Early users are giving it review and investigation work while Opus 5.5 keeps writing the code. The event also brought Codex Cloud 2.0, the Agents API and dots, and early benchmarks show the agent harness changing results.

GPT-6.1 Sol: early users give it review, investigation and computer use, and keep Opus 5.5 for writing code

OpenAI launched **GPT-6.1 Sol** at DevDay. Tibo (OpenAI) describes it as “near Astra intelligence at one fifth of the price of Astra” with a 95% cache-read discount [1]. Theo gives the prices as \$2/M input, \$10/M output and \$0.10/M cache reads. Cache reads used to be 10% of the input price, so this halves them. That matters because agent work is mostly cache reads [2]. When Opus 5.5 recalculated the cost of Theo’s past sessions at these prices, its estimate fell from \$5,700 to \$1,550 [2].

Theo’s position after testing: “I still think Opus 5.5 is the current GOAT for coding.” He doesn’t default to Sol for writing code, but does for code reviews, architecture analysis, computer use and email [3]. His workflow: - **Opus writes the code, Sol audits it.** On a T3 Code PR, Sol found two bugs that Fable and Opus had missed: follow-up messages could stay blocked after the agent started, and recovered setup progress disappeared too early. He pasted Sol’s findings into Claude to fix [2]. He calls Opus “a more pleasant collaborator” and says Sol is better at digging into details [2]. - **Let Opus call Sol.** He plans to let Opus 5.5 call Sol for root-causing bugs, triage and reviewing Opus’s own work, and already runs Sol inside Claude Code for this [2]. - **Don’t use Sol for long unattended rewrites or UI.** It ran for days on his TypeScript-to-Rust

compiler port without progress [4]. He says it “sucks at front end” and fills UIs with unnecessary text [2]. - **Use it to clean up after other agents.** In an audit of that port, Sol found that about 1.3M of 1.8M lines were unused. Opus had rewritten the port from scratch in a new crate and never deleted the old code [2].



OpenAI fights back (27:37)

The harness changes the benchmark result. Theo first reported Sol beating Opus 5.5 on Terminal-Bench 4 for about 1/30th of the price [5]. He then noticed that he had run it in Codex, while Artificial Analysis used mini-swe-agent [6]. Harbor Hub runs each model in its official harness; there every model scored higher than in Artificial Analysis’s runs, and Sol gained the most from moving to Codex [7]. When you compare models, run them in the harness you actually use.

Codex Cloud 2.0, the Agents API and dots

- **Codex Cloud 2.0** has configurable cloud environments. Tibo says that once you’ve configured one, “it’s impossible to go back to building on your laptop” [8]. It runs on the same infrastructure as dots [9]. At the keynote, Romain Huet sent “rewrite the entire back end in rust” to the cloud to check on later [10]. He also used an “app shot” to have Codex audit an app across screen sizes in the simulator, running in the background [10]. Simon Willison tried Codex Cloud for a phone-built live-blog tool, ran

into problems, and switched to Claude Code for web [11].

- **Agents API** (preview): the same technology that powers OpenAI’s cloud agents, with computer use [8].
- **Dots** are always-on agents running on Astra, with their own computer and browser. They are included in Pro without drawing on usage [12]. A Codex task that a dot creates is billed as normal usage [13]. Alexander Embiricos’s rule: use Codex when you want to do the work yourself, and a dot when you want to delegate [14]. He says Codex is like a principal engineer that won’t check Linear or Slack unless you explicitly tell it to [14].
- **Sign in with ChatGPT** lets you spend your included subscription usage inside partner products such as Devin and OpenCode [15]. Tibo gave “over 60” partners in one post and “over 16” in a later one [16].
- **Open models in Codex:** enterprise teams can run GLM-5.3 Flash and Kimi K3 natively, and the spend counts against their OpenAI commit [17].
- **Ultrafast** runs $8\times$ faster at $6\times$ the cost [9]. Matthew Berman says he burned \$1,000 in about 90 minutes with early access. He found that with inference this fast, local tool calls and terminal commands became the slow part [18].
- **Plans:** Plus is $1\times$, Pro 100 is $5\times$ and Pro 200 is $10\times$. Existing Pro 200 subscribers keep $20\times$ for a while and get extra credits [19]. Tibo says the reopened Pro \$200 plan works out to half the API-dollar value of the old one, with no five-hour limit [20].

Pruning tests written by agents

Peter Steinberger says OpenClaw deleted about 400k lines of its own tests “without much change in code coverage.” Models write tests for every tiny change, and his test-audit skill helped remove them [21]. Kent C. Dodds dropped about 70k lines of tests from Kody (PR #2707) [22].

Steinberger also moved his team’s agent sessions into one shared place. People were embarrassed for a day, and then learned from how differently everyone worked with their agents [14].

Sonnet 5.5: better as a subagent than as your main model

Theo’s advice on Sonnet 5.5 is to let Opus or Fable call it rather than prompting it yourself [23]. On his benchmark for planning how to land a huge PR, Sonnet scored slightly above Opus at about half the price, in about five minutes. That is codebase comprehension, not implementation [23]. On a game build it took 43 minutes against Opus’s 36 [23]. He recommends avoiding both max and low effort. In one benchmark, going from xhigh to max raised token use by 1,500% [23].

A reusable pattern for agents that change state

The Box/LangChain contract-review agent shares read tools with subagents. Tools that change state or wait for a human approval stay with the lead agent [24]. The rule: “The LLM extracts, the playbook decides.” Business rules live in code you can inspect and test [24].

Smaller items

- **Cursor** /`visualize` draws charts and diagrams inline in the Agents Window [25].
- **Pi**: MCP and codemode are shipped extensions, and you can turn them off with `pi config` [26].
- **LangChain Managed Deep Agents** has a terminal UI for test-driving agents locally: `mda chat` [27].
- **Kody** can run Codex agents [28]. Dodds says the Decisions API isn’t documented anywhere he could find yet, and suggests `jev` in the meantime [29]. His broader point is to keep integrations, memories and tokens outside any one agent so switching agents stays cheap [30]. He builds Kody, so he isn’t neutral on this.

Sources

1. X post by @thsottiaux
2. OpenAI fights back
3. X post by @theo
4. X post by @theo
5. X post by @theo
6. X post by @theo
7. X post by @theo
8. X post by @thsottiaux
9. X post by @embirico
10. OpenAI DevDay 2026 Keynote (FULL)
11. OpenAI DevDay 2026 live blog
12. X post by @thsottiaux
13. X post by @thsottiaux
14. LIVE from OpenAI DevDay
15. X post by @thsottiaux
16. X post by @thsottiaux
17. X post by @philipkiely
18. OpenAI COOKED
19. X post by @thsottiaux
20. X post by @thsottiaux
21. X post by @steipete
22. X post by @kentcdodds

23. OpenAI should be scared of this one
24. X article by @Box
25. X post by @cursor_ai
26. X post by @mitsuhiko
27. X post by @ndrezn
28. X post by @kentcdodds
29. X post by @kentcdodds
30. X post by @kentcdodds