Phone-First Agents, Open-Model Parity, and New Multi-Agent Surfaces

Coding Agents Alpha Tracker

2026-04-03

Phone-First Agents, Open-Model Parity, and New Multi-Agent Surfaces

By Coding Agents Alpha Tracker • April 3, 2026

Simon Willison's latest operating notes made the day's clearest point: serious coding-agent users are moving across phone, web, cloud, and parallel sessions instead of living inside one IDE. Also worth your time: Cursor 3, Juni CLI, Codex surface and pricing shifts, and strong new evidence that open models are now viable execution backends for coding agents.

TOP SIGNAL

The biggest practical shift today: **coding agents are becoming their own surface, not just an IDE feature**. Simon Willison says about **95%** of the code he produces is AI-generated and much of it happens from Claude Code on his phone; OpenAI says the **Codex App** is now its most-used surface; and **Cursor 3** now gives agents a separate collaboration window instead of hiding them inside the editor [1, 2, 3, 4].

> "Today, probably 95% of the code that I produce, I didn't type it myself." [1]

The operational implication is clear: high-end users are working across **phone, web, cloud, and parallel agent sessions**, then reviewing the results in GitHub or agent UIs rather than hand-typing everything in one local editor tab [1, 5, 6].

TOOLS & MODELS

- **Cursor 3** — ships a new agent interface as a **separate window** that complements the IDE. You can run agents locally, in a worktree, over remote SSH, or in the cloud; Cursor says cloud agents get **their own**

computers for autonomous work. It also recently launched **Composer 2** as a frontier model with high limits [3, 4, 5].

- **Cursor design mode** — now exposed behind + **+D** with click-to-edit, drag-to-draw, shift-drag boxing, and -click to send selected context directly to chat [7].
- **JetBrains Juni CLI** — Junie is now usable from the terminal. Jeff Delaney says install is a **single command**, he used it to build a dependency-risk analyzer, and he found it handled more complex tasks than other agents he had tried because of IntelliJ's deep project understanding. Junie also **routes between coding models automatically** [8].
- **Codex** — per OpenAI's Tibo, the **Codex App** has overtaken both the VS Code extension and CLI as the team's most-used interface. Business and enterprise pricing now starts at **$0 seats with pay-as-you-go**, and new business/enterprise users can get up to **$500 in credits** [2, 9].
- **Open models in Deep Agents** — LangChain's evals show **GLM-5** at **0.64 correctness (94/138)** versus **Claude Opus 4.6** at **0.68 (100/138)** and **GPT-5.4** at **0.61 (91/138)**; **MiniMax M2.7** scored **0.57 (85/138)**. The interesting part is not "open beats frontier" — it doesn't — but that open models are now **close enough on core agent tasks** to be viable execution engines, especially given the cost/latency gap [10].
- **Practical routing pattern** — Deep Agents CLI now supports runtime `/model` swapping, so you can use a **frontier model for planning** and then switch to a **cheaper open model for execution** mid-session [10].
- **Simon Willison's current model read** — he still defaults to **Claude Code** because it better matches his coding taste, but says **GPT-5.4** is now on par with **Opus 4.6** and cheaper, while **OpenAI Codex** is effectively comparable to Claude Code [1].

WORKFLOWS & TRICKS

- **Prompt the agent with `red green TDD`**
  1. Give the task.
  2. Explicitly say `red green TDD`.
  3. The agent writes a failing test first, runs it, then writes code until the test passes.
  4. Keep those tests in the repo so the next agent change has regression coverage too [1].
- **Start with a thin template, not a giant instruction file**
  1. Create a minimal skeleton repo in your preferred style.
  2. Include one tiny test — Simon uses a `1 + 1 = 2` test.
  3. Let the agent infer structure, formatting, and test style from the existing pattern [1].
- **Build a prototype bank the agent can recombine**

1. Save small tools and prior experiments somewhere durable.
2. Point the agent at old repos or markdown research notes.
3. Ask it to combine those working fragments into the new solution.

Simon does this with **193** HTML/JS tools in `simonwtools` and **75+** AI research projects in `simonw-research`, and explicitly values code that was **written and run**, not just generated as a report [1].

- **Phone-first ops work if the execution boundary is clean**
  - Simon uses **Claude Code for web** from the iPhone app against a GitHub repo, sometimes in **"dangerously skip permissions" / YOLO mode**, because the code runs on Anthropic's servers rather than his laptop; for important work, he reviews later in GitHub PRs [1].
  - Kent C. Dodds used **Kody via Claude on his phone** to inspect and bump memory on a failed deployment while at Disneyland. His setup keeps API secrets usable only for approved hosts and hidden from the agent itself [6, 11].
- **Use frontier/open model splits deliberately**
  1. Start with the stronger model for planning.
  2. Switch with `/model` to a cheaper open model for the repetitive execution work.
  3. Let the harness normalize context windows and tool-calling differences instead of hand-tuning each provider [10].
- **Optimize for interruptibility — but watch your cognitive ceiling**
  - Simon says the old "protect 2-4 hours of flow state" rule has changed: now he often needs **two minutes** to prompt the agent and then can move on to other work [12].
  - The catch: running **four agents in parallel** is productive, but it wipes him out by **11 AM**. The new bottleneck is often **human cognition**, not typing speed [12, 1].

PEOPLE TO WATCH

- **Simon Willison** — the highest-signal operator today by a mile. He is publishing agentic engineering chapters on his blog, sharing real production habits, and offering grounded model comparisons instead of demo-theater [1, 13].
- **Kent C. Dodds** — worth watching for practical "agent-from-your-phone" production workflows and a concrete pattern for **secret-scoped tool access** instead of raw credential exposure [6, 11].
- **Andrej Karpathy** — his latest workflow is a strong context-management pattern for technical research: ingest raw articles/papers/repos into `raw/`,

let an LLM compile a markdown wiki, use the agent for Q&A, run lint-style health checks, and file outputs back into the knowledge base [14].

- **Steve Yegge** — new **Beads v1.0.0** and **Gas Town v1.0.0** matter because they push on the hard parts of multi-agent systems: memory, orchestration, recovery, and a human-facing control surface that reduces the amount of reading users have to do [15].

WATCH & LISTEN

- **12:49-17:54 — Simon on dark factories.** He explains why "nobody types code" is already practical for him, then walks through StrongDM's next step: not reading code, and instead testing with swarms of simulated users and mocked APIs [12, 1].

An AI state of the union: We've passed the inflection point & dark factories are coming (12:40)

- **48:34-52:03 — Simon on phone-first coding and model choice.** This is the concrete setup segment: Claude Code web from an iPhone, GitHub repo requirement, YOLO mode, and why **GPT-5.4** has become a serious cheaper alternative to **Opus 4.6** [1].

An AI state of the union: We've passed the inflection point & dark factories are coming (48:33)

- **68:38-72:48 — Simon on `red green TDD`.** Cleanest short explanation today of why agents should always run code, write the failing test first, and leave behind regression coverage for the next session [1].

An AI state of the union: We've passed the inflection point & dark factories are coming (68:37)

PROJECTS & REPOS

- Beads — **v1.0.0** shipped today. It's a drop-in memory system / knowledge graph for coding agents; Beads crossed **20k GitHub stars** this week and now uses **embedded Dolt** for a more stable, versioned, SQL-queryable substrate [15].
- **Gas Town** — also hit **v1.0.0** today. Yegge says it has been stable for weeks after the Dolt migration, has **13k stars**, and is already being used by non-technical teams to build internal software, including a replacement for a niche SaaS product [15].
- Gas City — alpha successor to Gas Town. Key difference: it exposes the underlying primitives so you can build your **own orchestrators** with roles, messaging, cost tracking, multi-model dispatch, beads, patrols, and more; existing Gas Town configs can be imported directly [15].
- Deep Agents — open-source coding agent and CLI alternative to Claude Code. Current signal is less about stars and more about usefulness: LangChain is using it as the harness for **52-model evals**, including open-model testing and runtime model swaps [10].

Editorial take: the winners right now are treating agents like a cross-surface operating layer — phone, web, CLI, cloud — but the durable edge is still boring

software discipline: tests, templates, reusable context, and tight safety boundaries [1, 5, 11].

Sources

1. An AI state of the union: We've passed the inflection point & dark factories are coming
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 @kentcdodds
7. X post by @flowstated
8. He just crawled through hell to fix the browser…
9. X post by @thsottiaux
10. Open Models have crossed a threshold
11. X post by @kentcdodds
12. Highlights from my conversation about agentic engineering on Lenny's Podcast
13. X post by @simonw
14. X post by @karpathy
15. Gas Town: from Clown Show to v1.0