Stateful Agent Workspaces, Cursor 3, and Simon Willison's Claude Code Playbook

Coding Agents Alpha Tracker

2026-06-23

Stateful Agent Workspaces, Cursor 3, and Simon Willison's Claude Code Playbook

By Coding Agents Alpha Tracker • June 23, 2026

Today's brief centers on the shift from chat-based coding assistants to stateful agent workspaces. It includes Simon Willison's copyable Claude Code workflow, reusable memory patterns, and the most relevant releases from Cursor, Google, and LangChain.

TOP SIGNAL

Today's clearest pattern: good coding-agent work is getting less chatty and more infrastructural. Simon Willison's Moebius side-project worked because he staged the repo and weights, wrote `research.md`, made Claude Code maintain `notes.md` and `plan.md`, and iterated against real browser errors instead of treating the agent like a one-shot chat [1]. The same pattern showed up in releases: Cursor's cloud agents now get dedicated dev environments, Google is pushing Managed Agents plus Skills Registry, and LangChain is pushing stateful sandboxes/runtime execution so intermediate work stays out of model context and work can resume mid-session [2, 3, 4, 5].

TRY THIS

- **Front-load context, then force durable notes (Simon Willison).** In `/tmp`, clone the target repo, weights, and likely helper libs; use a separate model session to produce `research.md`; `git init` a clean project; then start Claude Code one level above it with `Read ./moebius-web/research.md...` plus a follow-up telling it to commit early/often and maintain `notes.md` + `plan.md` [1]. Ask early for `the URL I can visit in my own browser`, paste errors/screenshots back in, and only then publish via `hf` CLI + GitHub Pages [1]. When a reference

project is ugly or obfuscated, offload inspection to a subagent; Simon used that to discover the `caches.open('transformers-cache')` pattern for ~1.3 GB browser caching [1].

- **Make each good run compound.** Jason Zhou's loop template keeps a shared artifact/knowledge layer plus logging and verification so the next run starts from more than chat history [6]. Simon uses `notes.md` for the same reason [1], and Kent C. Dodds' cleanup prompt is excellent: ask what was learned that would be repeatable and useful for future agents, then have the agent update documentation before you end the session [7].
- **Have the agent recombine working systems instead of inventing from scratch.** Simon Willison says he gets the best results by asking coding agents to combine existing projects rather than build from a blank prompt [8]. In practice, he runs Claude Code against a real repo, lets its explore agents inspect the README and adjacent plugins, then asks it to generate the new plugin and run tests [8].
- **Define the win condition and the verification method up front.** Thibault Sottiaux says long-horizon agents make bad assumptions when you only say 'optimize this'; tell the agent whether you want something like a ~20% gain or a 14x gain, and specify how it should verify the result against production-like workloads or log replay [9]. For riskier automation, pair autonomy with a second agent: Codex's auto-review/Guardian checks each action against the original intent, prompts on medium risk, and blocks high-risk actions [9].

WHAT SHIPPED

- **Cursor 3:** Michael Truell says 95%+ of users now use Cursor primarily as an agent; agents are used ~5x more than assistive features on a request basis and far more by lines of code [2]. New agent-first surfaces include design mode, recursive subagents, and cloud agents with dedicated dev environments for tests, screenshots, and interactive review [2]. Cloud agents are now 3x faster with 99.9% reliability; Automations has already seen 6M+ runs, including Amplitude's background migration of 20k React components to Tailwind [2].
- **Cursor Mobile + Origin:** iOS beta lets you kick off/review agents, inspect artifacts/screenshots, annotate issues, and remote-control local agents [2]. Origin is Cursor's agent-native Git layer: it can resolve merge conflicts, fix CI failures, handle PR comments, and claims review-time reductions of more than 50% [2].
- **Google's agent stack:** Addy Osmani framed the current stack as four rungs: Agent Studio, Managed Agents API, Anti Gravity 2.0, and ADK 2.0 GA [3]. Also notable: Agent CLI for scaffold/run/eval/deploy, Skills Registry for org-scoped markdown skills with dynamic discovery, and

Gemini 3.5 Flash as the new default for long-horizon agent work [3].

- **LangChain:** Deep Agents v0.6 adds a code interpreter so tools can run inside the runtime, intermediate results stay out of model context, and only relevant output comes back — fewer round trips, less token waste [5]. `dcode` is the provider-agnostic coding agent layer for trying OSS models like GLM 5.2; docs: deepagents code overview [10]. LangChain is also explicitly pushing stateful agent computers; see LangSmith Sandboxes [4, 11].
- **Open-source artifacts worth cloning:** Jason Zhou open-sourced the loop-engineer-template with shared artifacts, logging, verification, and a harness for compounding work across runs [6]. Simon Willison shipped a live Moebius browser demo after using Claude Code to port the model to ONNX for in-browser use [12, 13].
- **Comparison worth noting:** Peter Steinberger says a complex three.js Rocket League-style task burned through a 5-hour usage allowance in under one prompt and still needed 7-8 fix rounds; he says GPT 5.5 handled the same task without follow-ups, which left him skeptical of multi-model routing [14, 15].

GO DEEPER

- **5:51-7:12 — Thibault Sottiaux on the difference between 'better ChatGPT' and a real engineering agent.** The practical point: connect the agent to Slack, Datadog, logs, and company systems, or you are leaving most of the value on the table [9].

The New Shape of Engineering (5:51)

- **23:16-25:17 — Michael Truell on where Cursor's next model is aimed.** Good framing if you think code generation is no longer the bottleneck: the target is broader engineering work like tool use, planning, testing, and UI around showing what changed [2].

Opening Keynote | Compile 26 (23:16)

- **29:58-34:20 — Thibault Sottiaux on trust after line-by-line human review stops scaling.** Strong segment on replacing blanket review with tests, log replay, end-to-end checks, and better observability [9].

The New Shape of Engineering (29:58)

- **Study Simon Willison's working files, not just the finished demo.** Start with research.md, notes.md, understanding.md, and the full Claude Code transcript. It is rare to get the full prompt -> plan -> notes -> deploy trail in public [1].
- **Study the loop-engineer-template.** If you want a minimal skeleton for shared artifacts, logging, and verification, this is one of the cleaner public starting points from today [6].

Editorial take: the edge is moving from clever prompting to better agent infrastructure — stateful workspaces, durable memory, tool connections, and explicit verification are what make runs compound instead of reset. [2, 3, 4, 6, 7]

Sources

1. Porting the Moebius 0.2B image inpainting model to run in the browser with Claude Code
2. Opening Keynote | Compile 26
3. Google's new AI agent stack from I/O 2026
4. X post by @LangChain
5. X post by @LangChain
6. X post by @jasonzhou1993

7. X post by @kentcdodds
8. Simon Willison - “How can we use LLMs and coding agents for data journalism?” (Story Discovery 2026)
9. The New Shape of Engineering
10. X post by @sydneyrunkle
11. X post by @LangChain
12. X post by @simonw
13. X post by @simonw
14. X post by @LLMJunky
15. X post by @steipete