Spawn drops PRs and ships every hour with a fleet of Fable agents; LangChain gives its agents a browser for end-to-end tests

Coding Agents Alpha Tracker

2026-10-30

Spawn drops PRs and ships every hour with a fleet of Fable agents; LangChain gives its agents a browser for end-to-end tests

By Coding Agents Alpha Tracker • October 30, 2026

A detailed case study of a fully agent-run deploy loop, LangChain's browser and async-subagent release, T3 Code's multi-harness orchestration, and the economics of OpenAI's new Codex limits.

Spawn: no PRs, an hourly ship loop, and agents that read production

The most extreme agentic-engineering setup of the day comes from Spawn, a game platform. Simon Willison wrote it up and said his eyebrows went up "at least a dozen times." By founder Jacob's account, a swarm of Fable-powered agents builds the product. The tokens cost over $3M a month, and Jacob says "we spend 30x payroll on AI tokens" [1]. If you are thinking about more autonomous pipelines, these are the parts worth studying:

- **No PRs.** Agents work directly on the code, the live product and the database. Changes that aren't quite right are allowed in, as long as everything is consistent once an hour. When something breaks, a new agent is created on the spot to fix it on top of the change, rather than undoing it [2].
- **Verification is split up.** Before landing a change, an agent runs the product and checks the change by using it. The full test suite runs against the latest code every 15 minutes. Helper agents that didn't write the change try to break it. Anything irreversible, such as payments, goes

through "dumb, hard-coded gates" [2]. They report that 1,055 of 123,548 changes were reverted over the past year, about 0.9% [2].

- **Production drives the work.** About 60% of work starts from production errors, bug reports filed by Spawn's in-product AI (Savi), log anomalies or feature requests. A fix usually reaches the affected user within an hour [2]. Everything is built so an agent can follow it step by step in text and replay it later [2].
- **An hour without a ship counts as a SEV-0.** Their reasoning is that an agent can only keep a few hours of work in its head, so shipping faster keeps it coherent. Rollbacks take 30 seconds, and alerts start or wake an agent that owns the incident [1].

Model choice: Jacob says Fable 5/5.1 made this level of autonomy possible and Opus 5.5 "doesn't quite hack it." GPT-6 Astra, asked to apply fixes from a security review, turned off every feature it judged insecure [1]. Willison doubts the approach would carry over to software where failures cost more [1].

LangChain Managed Deep Agents v0.10: browser end-to-end tests and background subagents

Version 0.10 is in public beta. The parts most useful for coding agents:

End-to-end UI testing. To add a browser, export it from `browser.py` at the project root: `define_browser(scope="thread", idle_ttl_seconds=600)`. Thread scope gives each conversation a fresh browser. `scope="user"` shares cookies across a signed-in user's threads [3]. The agent starts the app with `execute`. A private `localhost` bridge lets the browser reach it, for example at `localhost:5173`, without a public URL. The agent then clicks through real flows [3]. Each `browser_run` call is a piece of an ordinary Playwright script. Once a test passes, the agent can save the steps as a test file, and later runs replay it with no model calls until the page changes [3]. LangChain uses this in its Open SWE coding agent [3].

Tools and cost. `browser_snapshot` labels elements from the accessibility tree. `browser_run` batches actions or code in one call, which keeps page content out of context. `browser_handoff` lets a person sign in without the password reaching the model or traces [3]. On 100 WebVoyager tasks, LangChain reports a 90% pass rate for $1.98 in total. The next-best of the seven toolsets it tested, Browserbase's MCP server, passed 89% at 9× the cost [3].

Async subagents. Add `"async": True` to a subagent definition. The orchestrator keeps talking to the user and can start, check, update or cancel the task [4]. Subagents send a notification when they finish, so the agent doesn't sit in a polling loop. They also share the agent's sandbox files [4].

T3 Code: one thread fans out across harnesses and machines

Theo's new walkthrough video shows how he works: - **Multi-model review.** In one thread he asks it to start four delegates (GPT-6.1 Sol, Muse, Gemini 4 Argon in Antigravity, Sonnet 5.5 in Claude) to explore the repo and find easy wins. Each runs in its own harness on his existing subscription, inside a worktree that T3 Code creates [5]. Threads can start threads in other repos and message each other through the orchestration MCP [5]. - **Remote fleet.** `npx t3 serve` exposes an instance over WebSocket. Tailscale or T3 Connect (free for three devices) makes it reachable remotely. You choose a machine per thread, or let load balancing pick the one with the most free CPU and memory [5]. Theo says he rarely runs threads on his own laptop anymore [5]. - **Treat the sidebar as an inbox.** You "settle" threads instead of archiving them, and can snooze them. Cmd+Enter sends a prompt in the background. A thread settles automatically when its linked PR merges [5].

Control of the whole fleet through one MCP server is now in the stable release. A button switches every machine from nightly to stable [6]. The thread detail panel can now show your usage limits and context usage [7].

Plan economics: what limits are worth, and how to compare models

OpenAI doubled usage limits for its Pro 100/200/500 and Business Premium plans. Dot usage stays separate from default usage [8]. A Codex app lead also announced a sprint to improve the app's simplicity and performance [9]. Theo estimates the $500 plan is now worth about $10.5k of usage a month and the $200 plan about $4.3k [10]. Separately, the $100/$200/$500 plans now include the same amount in API credits for one API organization, which matches Anthropic's earlier offer [11].

Before choosing a model based on those numbers, Adam Holter warns that API-equivalent value can mislead. On $20 plans, he says "$1 of Sol costs 7x what $1 of Opus does. A task, only 1.3x" [12]. He also notes that none of the models he charted rank the same on tokens per second as on time per task [13].

Short notes

- **Antigravity** launched a plugin marketplace with one-click install and auth for Notion, Stripe, Linear, Google Docs and others [14]. Jules is winding down, and Google is bringing its remote background-task execution into Antigravity [15].
- **Era alignment benchmark:** 19 models ran 2,304 times across 41 tasks in simulated companies and broke security or business rules in 40% of runs. Every model failed at least once [16]. The authors advise testing

each agent before it gets a login and keeping your own record of what it changes [16].

- **Kent C. Dodds** is building Kody "profiles": tokens or MCP connections limited to a subset of your account's permissions, which makes them safer and auditable [17].
- **swyx** says an app-design skill is the most valuable addition he has used with any model for building real working software [18].

Sources

1. Building a multiplayer game world with autonomous Fable-class models
2. X article by @jsnnsa
3. Browser Use in Managed Deep Agents
4. Managed Deep Agents v0.10: Browser use and asynchronous subagents
5. T3 Code broke me
6. X post by @theo
7. X post by @theo
8. X post by @thsottiaux
9. X post by @ajambrosino
10. X post by @theo
11. API credits benefit for eligible ChatGPT subscriptions
12. X post by @AdamHoltererer
13. X post by @AdamHoltererer
14. X post by @kevinhou22
15. X post by @julesagent
16. X post by @OfirEhrlich
17. X post by @kentcdodds
18. X post by @swyx