Claude Code team's verification playbook, and practitioners split on GPT-6.2 Astra

Coding Agents Alpha Tracker

2026-11-18

Claude Code team's verification playbook, and practitioners split on GPT-6.2 Astra

By Coding Agents Alpha Tracker • November 18, 2026

How the Claude Code team verifies agent-written code: a repo-local verify skill, a three-round stop rule, and checks that trigger on the diff. Plus a backlash from early testers against GPT-6.2 Astra, Sol's quieter reception, and new tools for coordinating agents from Cursor and Cognition.

The Claude Code team's verification rules

In an article Addy Osmani shared, the Claude Code team says about 90% of its code is written and reviewed by Claude. It argues that without verification loops you can supervise only one or two agents, because you check everything yourself, and with them you can run "dozens or hundreds in parallel" [1]. The practices:

- **A `verify` skill.** Put it in `.claude/skills/`, as close to the relevant code as possible. It should boot and drive the real app: the Claude Chrome extension for web, an iOS/Android MCP for mobile, computer use for desktop, or a local service for backend. Claude uses it before committing. The team treats it as the smoke-test harness and asks Claude to stress-test it until it is fast and reliable [1].
- **Test types worth investing in:** end-to-end first, because they're "difficult to fool" [1]. Then property tests with fast-check, Hypothesis or proptest, checking round-trips, invariants, idempotency and equivalence [1]. For migrations, differential tests run the old and new systems on random or sampled production inputs and keep a list of approved differences [1]. Flaky tests teach the agent to rerun rather than fix [1].
- **Three-round stop rule.** If a check still fails after three fix attempts, Claude stops and brings in a human [1].

- **Checks that trigger on the diff.** A 39-item reliability checklist records, for each item, a trigger readable from a diff, a check, and the incidents or PRs behind it. Claude runs only the matching items, usually 3–10 [1].
- **Gates the change can't edit.** REVIEW.md is read from the base branch, not from the PR [1]. Tests, build and linter run before a PR opens, enforced with a hook or a `/goal` [1]. To stop repeat failures, add a hook or a CLAUDE.md rule [1].

Two other items this period support the property-testing advice. In a paper shared by @eulerfx covering 25 agent-built apps, properties derived from the spec found 56 bugs that properties derived from the code missed, and the code-derived properties found 43 the spec-derived ones missed. Spec-derived properties mostly caught state, lifecycle and logic errors; code-derived ones mostly caught input handling [2]. Caveats: the bugs were judged by the same model that wrote the tests, the prompts differed between the two approaches, and the method needs a good spec [2]. In a Bend video, ThePrimeagen recommends writing the laws yourself and having the agent write the proofs. He notes that agents often change unit tests along with the code, which "doesn't give me confidence." His suggested target is a job-ticket state machine [3].

GPT-6.2 Astra and Sol: cheaper Astra, mixed reviews

The models are in the API as `gpt-6.2-astra` and `gpt-6.2-sol`. Astra costs $8/$40 per million input/output tokens, and cached input drops to $0.40 from $1, which Simon Willison calls good news for long agentic sessions. Sol stays at $2/$10 [4]. Willison supports both in LLM 0.38 [4], and Armin Ronacher says Pi users can get them with `pi update --models` [5]. On Willison's Three.js/OpenStreetMap mech test at Ultra effort in Codex CLI, neither model matched Claude Fable 5.2 or Gemini 4 Carbon [4].

Theo, who tested Astra early, would still pick Opus 5.5. He credits Astra with strong review, 3D/game work, computer use and subagent prompts, plus a reverse-proxy design he says will save $25k a month [6]. His failure reports are useful for anyone running long jobs: - **It blocks itself on questions you can't answer.** Two hours into a `/goal`, it stopped to ask about a reviewer's clip filenames. He saw the same in Codex and now runs agents on crons to check whether it has blocked again [6]. - **Its summaries are hard to read.** He switches to Opus 5.5 at the end of a long run just to get a readable summary [6]. He also uses Opus as an "auditor and translator" for Astra through a T3 Code feature [7]. - **It runs without progress.** On his `ts-rust` benchmark, Opus finished in under two weeks for about $20k in tokens. Astra ran seven days, landed no improvements and made the compiler 56% slower, then ran four more days with no changes after he redirected it [6].

Other views: Tibo (OpenAI) says early testers' videos predate "a lot of changes in the last 3-4 days." He recommends Sol ultrafast, now on Pro 200 plans as well as Pro 500 [8, 9, 10]. Theo has heard, secondhand, that many OpenAI

engineers use Sol Ultrafast rather than Astra [11]. swyx disagrees with the negative reviews: he calls Astra the best workflow-automation model he has used, "no longer getting stuck in horrible loops," and says it runs much of Latent Space's publishing pipeline [12].

Coordinators and persistent agents

Cursor Projects. You talk to one coordinator, which starts workers, for example on the data layer, the API and the UI. When one worker changes something like a schema, the coordinator pushes the change to the others [13]. In multiplayer projects, coordinators message each other to split work and avoid conflicts. Cursor says this makes it 5–10x more productive internally [13]. The speaker's setup advice: - Connect docs, the task tracker, Slack and monitoring data [13]. - Require a demo video in the PR for UI changes, and have the agent watch CI and fix Bugbot and security-review comments [13]. - Configure a cloud environment so the agent can actually use what it built. The speaker calls this the single biggest lever on dev velocity [13]. - Route reviews by risk: schema changes need a human, flagged UI changes may not [13]. - Projects work best with 3–8 people and a medium-term scope [13].

Devin Bots give each project its own persistent identity and memory, replacing short-lived sessions [14, 15]. Cognition's Steven K says this means you should delegate "goals and responsibilities," not tasks. He adds that the change was "almost entirely an orchestration change" that could probably have been built a year ago [15]. After a week of trials, swyx handed over QA, cross-platform app porting and video editing. His advice is to expect agent managers that own outcomes, not "AI Employees" [16].

Also notable

- **Claude Code Code Mode** (experimental): Claude writes scripts that chain your tools together and can reply while long commands are still running. Opt in with `CLAUDE_CODE_EXPERIMENTAL_CODE_MODE=1` [17].
- **Sandbox placement:** LangChain says calling the sandbox as a tool lets you update the agent without rebuilding an image and keeps API keys outside the sandbox. The cost is network latency [18]. Only microVM-style kernel isolation stops a compromised agent from exploiting kernel bugs [18].

Sources

1. X article by @addyosmani
2. X post by @eulerfx
3. finally a language for vibecoders
4. GPT-6.2 Astra and GPT-6.2 Sol

5. X post by @mitsuhiko
6. X article by @theo
7. X post by @theo
8. X post by @thsottiaux
9. X post by @thsottiaux
10. X post by @thsottiaux
11. X post by @theo
12. X post by @swyx
13. Coordinating Agents on Large-Scale Work with Cursor Projects
14. X post by @cognition
15. X post by @stevenkplus1
16. X post by @swyx
17. X post by @ClaudeDevs
18. How to Choose the Right Sandbox for AI Agents