

Verification loops over line review: Osmani’s ”define done” and Sanfilippo’s bit-identical emulator port

Coding Agents Alpha Tracker

2026-10-12

Verification loops over line review: Osmani’s “define done” and Sanfilippo’s bit-identical emulator port

By Coding Agents Alpha Tracker • October 12, 2026

Practitioners argue that agent output quality comes from checks the agent can run itself. Also covered: GPT-6.1 Sol’s default context in Codex rises to 500k, T3 Code nightly adds /goal, and Theo’s case for judging models by cost per completed task.

Quality comes from checks the agent can run

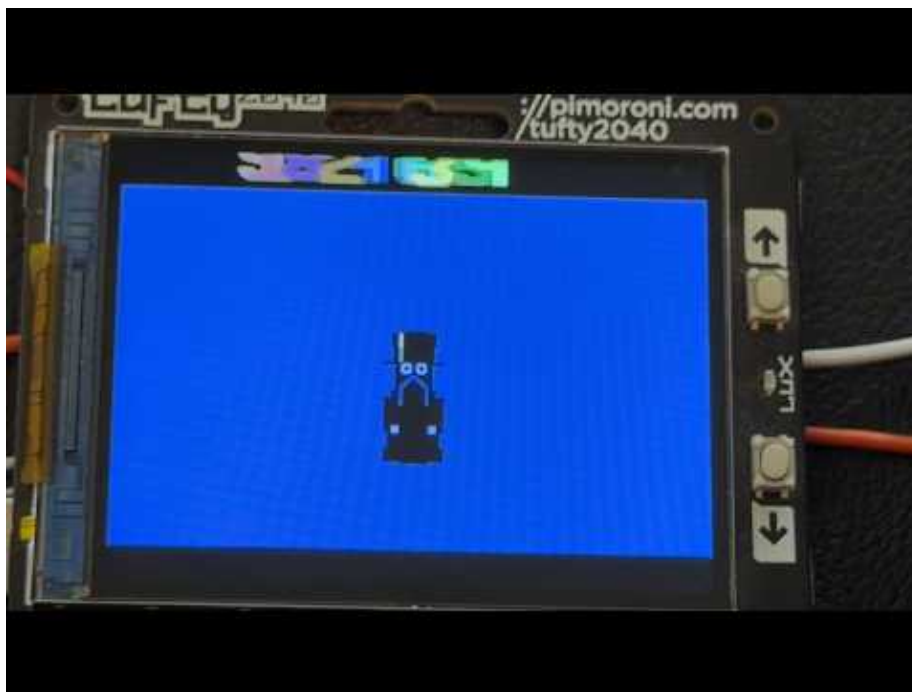
Addy Osmani made the clearest argument this period. He says agents write better code “at scale” because they make checking cheap. Tests, benchmarks, specs and real end-to-end runs used to be what made good code expensive, and now you can ask an agent for them. So the job shifts from “review every line” to “build loops the agent can run itself” [1]. He also says to make taste compound: when the agent gets something wrong, write the fix into shared config, a hook or a test so it doesn’t happen twice. You still own what ships, but you review evidence as well as diffs [1].

He was responding to a complaint that asking Claude to review code is a slot machine that finds new bugs every time. His fix for that has three steps [2]:

1. Define “done” before review starts. Without that, a reviewer (human or agent) can always find something to improve.
2. Give Claude checks it can run, such as bash commands, tests or clicking through the app in a browser, and require them to pass before it says it’s finished.
3. When nothing blocks, merge and ship.

Salvatore Sanfilippo (antirez) shows the same idea on hardware, in a video where Opus 5.5 reworks his RP2040 ZX Spectrum emulator and adds a Commodore 64 emulator. The video is in Italian. Several steps are worth copying:

- **Make a separate scratch copy first.** Before porting his C64 emulator into the device project, he had the agent make a gitignored, uncommitted copy inside the repo. It converted that copy to instruction-at-a-time execution on its own and reported back before any integration. He warned it would get tangled if it converted and ported at the same time [3].
- **Use the old implementation as the oracle.** The agent ran the old cycle-accurate emulator and the new one side by side, requiring matching images and registers. After 10,000 frames the output was still bit-identical [3].
- **Measure, then redesign.** The faithful port ran at about 10% of real speed. The bottleneck was the VIC video chip [3]. He told the agent to capture register writes and render frames from them instead of emulating the chip cycle by cycle [3]. The result runs Spectrum and C64 in one device at 400 MHz, including timing-heavy C64 demos at 50 FPS with sound [3].
- **Give the agent instrumentation and inputs.** It simulated key presses itself [3]. He says its first port worked on the device first try [3], and its instrumentation checked for delays, dropped frames and audio problems [3]. His C64 emulator also exposes a direct API for stepping, debugging and grabbing frames, which he prefers to an MCP layer he sees as adding friction [3].



Miglioramento di ZX2040 attraverso Opus 5.5: speedup e emulazione del Commodore 64 (37:50)

Separately, **MaciekCodes** reports that his own coding agent, “a ‘while loop’ and a few tools,” improved itself quickly [4].

GPT-6.1 Sol: bigger default context in Codex, and plan math

Tibo (OpenAI) says GPT-6.1 Sol’s default context in Codex & Work went from 272k to 500k, about 80% more. The aim is fewer compactions and better handling of long, nuance-heavy tasks. You can set it higher or lower, up to Sol’s 1M maximum [5]. Subscription usage of Sol carries no long-context fees [6].

Theo thinks the default speed was raised too much and is now barely slower than “fast” [7]. He estimates he could use Sol Ultrafast as his main model on the \$500 plan and only just hit his limits [8]. That figure is his total usage (583/week), and it leaves out his long-running ts-rust work [9]. He also built a dashboard that tracks Claude usage across his accounts to roughly predict how long his limits will last [10].

Theo: judge models by cost per completed task

In a new video, Theo argues that per-token price is the wrong measure for agentic work. Token-hungry models can wipe out a low sticker price. For

agentic dev work he says caching, reliability and token efficiency matter much more [11]. He says bills are rising because agents now run longer and finish bigger tasks [11]. He favors GLM 5.3 Flash for real-world agentic work, citing its vision, tool calls and long-running work [11].

The sponsored segment of the video (Railway) still describes a usable pattern. On his Lakebed project, agents set up production, staging and per-PR environments for him. Claude or Codex connects through Railway’s MCP, spins up an environment, verifies a change, ships by merging to main, then checks on the environment afterward [11].

For local compute limits, he moved ts-rust’s build and test commands onto Blacksmith runners. Each thread still takes a while, but he is far less bottlenecked when running many in parallel [12].

T3 Code nightly

Theo added a proper `/goal` flow with a visual callout to the T3 Code nightly [13]. A user showed off the new inline rendering in chat by asking the agent whether a monitor mount would clamp onto their desk [14]. Theo says he hadn’t expected the feature to be so useful [15]. For hazy features he still uses Matt’s “grilling” skill, noting that modern models are good at asking questions [16].

Smaller notes

- **Riley Brown** shared a `grokbot-webhook` skill. Claude Code hands X/Twitter research to GrokBot, waits for the answer, then continues with the task [17]. He says it works with any agent, including Codex [18].
- **Kent C. Dodds** shared the Kody v2026.10.11 release (release notes). Cron jobs, webhooks, subscriptions and inbound email now run with the permissions of whoever enabled them, checked live on every run, with no fallback to the org Owner [19]. His caption: “This is what 122 PRs in one day looks like” [20].
- **Armin Ronacher** says no single agent runtime wins. VMs, sandboxes, durable objects or a plain JavaScript interpreter each fit different situations [21].

Sources

1. X post by @addyosmani
2. X post by @addyosmani
3. Miglioramento di ZX2040 attraverso Opus 5.5: speedup e emulazione del Commodore 64
4. X post by @MaciekCodes
5. X post by @thsottiaux
6. X post by @thsottiaux

7. X post by @theo
8. X post by @theo
9. X post by @theo
10. X post by @theo
11. I thought AI was gonna get cheaper?
12. X post by @theo
13. X post by @theo
14. X post by @aflekkas_
15. X post by @theo
16. X post by @theo
17. X post by @rileybrown
18. X post by @rileybrown
19. X post by @kodykoala
20. X post by @kentcdodds
21. X post by @mitsuhiko