Turn Agent Feedback Into Project Memory

Coding Agents Alpha Tracker

2026-07-26

Turn Agent Feedback Into Project Memory

By Coding Agents Alpha Tracker • July 26, 2026

Today's strongest coding-agent pattern is cumulative feedback: preserve corrections as searchable project guidance, then pair parallel execution with bounded scope and independent review. Also covered: signed-in Work browsing, OpenClaw autoreview, and practical Opus 5 routing signals.

TOP SIGNAL

Make agent feedback cumulative. ThePrimeagen records every agent exchange and commit, then has an agent distill the history into a searchable project `field guide/`; after 10 sessions, he reports that deep feedback fell sharply without changing the model or harness. [1] Peter Steinberger's release-prep QA loop applies the same principle at larger scale: bounded parallel work, root-cause fixes, and a continuously updated test report instead of repeated ad-hoc prompting. [2]

TRY THIS

- **Create a retrievable field guide from your corrections.** Start by being exact about the code you want—ThePrimeagen uses voice dictation and nitpicks output. Save every back-and-forth plus the resulting commit hash in Markdown. Periodically ask an agent to review that record and write *durable, generalized* guidance into `field guide/`, with an `init.md` containing links and short descriptions. Let future agents tool-search that directory rather than loading the entire history into context. [1]
- **Run parallel QA with explicit non-negotiables.** Steinberger's OpenClaw release-prep request: use **12 subagents** split by functionality; run dev gateways on separate ports and reserve some for stress tests; use worktrees and create PRs autonomously; target **200 bugs**; fix root causes rather than patches; allow refactors except across the plugin-SDK boundary; and keep a Markdown test report continuously updated. [2] This is

a replicable pattern: parallelize discovery, but constrain the fix scope and require an auditable artifact.

- **Make planning a separate, reviewable phase.** Theo's workflow is to run Claude Code with Opus 5 at high effort to explore a codebase and propose a plan, have another model independently review that plan, then revise based on the critique. For work intended to merge, he plans to use Opus for implementation and ask Fable and/or 5.6 SOL for a thumbs-up/down review. [3]
- **Feed lint output directly to an agent.** Simon Willison notes that Ruff v0.16.0's output contains everything a coding agent needs to fix reported problems. Use the linter output as the handoff artifact, then have the agent make and verify the fixes rather than translating errors into a separate prose prompt. [4]

WHAT SHIPPED

- **ChatGPT Work agent: signed-in cloud browsing.** You can take over its cloud browser to authenticate on a site, then return control to the agent; the login persists across sessions. Tibo reports the capability is already available in the ChatGPT app. [5, 6]
- **OpenClaw `autoreview` skill:** Steinberger reports a record **66 review rounds** on a complex refactor. The skill definition is public at openclaw/agent-skills. [7]
- **Opus 5: strong practitioner results, but watch fit and usage.** Theo calls it the first Anthropic model he has used that follows instructions without repeated constraints, and reports using roughly 20–25% lower cost per task than Fable in his own work. [3] But reports are not uniform: Armin Ronacher says he likes Opus 5 while finding it expensive and token-hungry, while another early tester reported that existing skills and plugins had to be rebuilt after it argued with instructions and stopped early. [8, 9] Treat a model upgrade as a harness regression test, not a drop-in swap.
- **High-velocity production signal:** Kent C. Dodds reports **135 merged PRs** in kodykoala since Monday, using Cursor with Fable, Grok, and GPT models plus CodeRabbitAI. [10, 11]

GO DEEPER

- **29:20–31:52 — Theo on where Opus 5 fits in a coding stack.** A candid model-routing discussion: Opus's diligence versus Fable's taste and SOL's tendency to overproduce code, plus the rationale for independent re-

view. [3]
Opus 5 is my new go-to model (29:20)

- **20:16–21:52 — Record a workflow; turn it into a skill.** Riley Brown demonstrates Claude desktop recording screen actions and converting them into a reusable skill—an alternative to writing brittle step-by-step instructions. [9]

Opus 5 Is Here… But NEW Claude Voice Is Even Bigger (20:16)

- **Repo to study: OpenClaw's `autoreview` skill.** The public definition behind a 66-round refactor review is a useful reference for designing persistent critique loops. Read the skill. [7]

Editorial take: the durable advantage is not a single model—it is a workflow that converts failures, reviews, and tool output into compact project memory for the next run. [1, 2]

Sources

1. X post by @ThePrimeagen
2. X post by @steipete
3. Opus 5 is my new go-to model
4. Ruff v0.16.0
5. X post by @OpenAIDevs
6. X post by @thsottiaux
7. X post by @steipete
8. X post by @mitsuhiko
9. Opus 5 Is Here… But NEW Claude Voice Is Even Bigger
10. X post by @kentcdodds
11. X post by @kentcdodds