

How the Claude Code team reviews agent-written code: more than twice the PRs merged, over 40% fewer reverts

Coding Agents Alpha Tracker

2026-10-26

How the Claude Code team reviews agent-written code: more than twice the PRs merged, over 40% fewer reverts

By Coding Agents Alpha Tracker • October 26, 2026

Anthropic’s Claude Code team describes its layered, mostly automated review pipeline and the limits it puts on review loops. Also: Theo uses Opus to fix T3 Code’s UI, Gemini makes things up in agentic review, and LangChain shows how its engineering agent handles identity.

The Claude Code team’s review pipeline

Addy Osmani shared a write-up of how the Claude Code team reviews code. Claude writes about 90% of Claude Code. From June to September, merged PRs more than doubled while the share of PRs that had to be reverted fell by more than 40% [1]. The team still reviews code. Review now happens continuously while the code is written, so Claude has usually reviewed and corrected a change at least once before a human looks at it [1]. Their goal is for most PRs to merge with at most one person in the loop, and they treat revert and incident rates as a guardrail [1].

Patterns you can copy:

- **Ask for proof before opening a PR.** Have Claude run something, such as tests, the build, or a screenshot compared with the design: “Instead of saying the task is done, we ask it to show us the result.” Then a subagent with fresh context reviews the diff. Many team members also run their own skills for bug hunting, shrinking the diff, end-to-end validation and checking UI against design guidelines [1]. Boris Cherny estimates a verification process improves output quality by 2–3× [1].

- **Fix repeated findings at the source.** If the reviewer keeps flagging the same issue, change the skill, add a lint rule, or update the docs. Keep `CLAUDE.md` short, since it is loaded into every review. In `REVIEW.md`, owners say what counts as failure, limit nitpicking, list paths to ignore, require file and line references, and ask for a top-level summary that includes a count and a clear “no issues” when there are none [1].
- **Filter findings with several agents.** Agents hunt for bugs in parallel. A separate vote keeps only findings with enough support, and findings that are only about style or cleanup are dropped. The team A/B tested these filters on its own PRs. Comments are labeled Important, Nit or Pre-existing [1]. These features ship as Code Review for Claude Code on Team and Enterprise plans, with a lighter local `/code-review`. Since the preview, the share of internal PRs getting substantive comments rose from 16% to 54% [1].
- **Cap review loops.** One PR went back and forth between agent author and agent reviewer more than 50 times overnight. About 59% of what the reviewer kept flagging was code that had been there since the first pass [1]. The fix: after three cycles, the reviewer asks the author to rethink the approach. Later rounds look only at the diff, name the class of bug, and don’t repeat findings already reported [1].
- **Set human review by the change’s risk, not the author.** Classifiers decide the risk level. Sensitive paths and CI workflow files always need another approval. Agent approvals don’t count as human approvals, and approval must be given again after untrusted commits [1].
- **Use past incidents as the eval set.** The review system is tested daily against a collection of PRs that caused incidents. Noise is measured by how people respond to each comment [1].

Humans still decide whether to build something, audit dangerous code like billing and CI, review architecture, check the evidence, and answer for every change [1]. On the same theme, Theo posted: “If you’re still merging most PRs by hand, you’re behind” [2]. Armin Ronacher polled whether people still write more than 20% of their code by hand. He reads the results so far as showing the base is “divided” [3, 4].

Theo: Opus fixes UI, and the hallucination record of another model

Robin Ebers criticized T3 Code’s fonts, border radius, icons and font hierarchy, and pointed to Capy as inspiration [5]. Theo asked whether he could turn the UI into “a hill climb for Opus” [6]. He opened `pingdotgg/t3code#24837` [7], later posted “Holy shit it worked” [8], and called Opus 5.5 “the greatest model of all time” [9]. After a few hours on the build, he said he hadn’t realized how much better a cleaned-up design would feel [10]. If you have a vague design complaint, try turning it into a goal the model can iterate toward.

He also ran a thread on an Ubuntu machine that drove an iOS simulator on a

Mac mini connected through T3 Code. He calls it “the best iOS dev experience I’ve ever had” [11].

Theo’s council skill has models review each other’s work. He now lists specific Gemini hallucinations from one run: a line number (line 173 cited in a 72-line file), IP addresses, command arguments, and function exports and returns [12]. He calls it “a straight up liability in your stack” [13]. These are his own observations, not a benchmark.

LangChain’s OpenSWE: identity and sandbox design

LangChain’s engineering agent, OpenSWE, drafts PRs and reviews every PR [14]. When you message it, it acts as you and opens PRs under your GitHub handle. When it reviews in the background, it uses its own GitHub App identity [14]. It picks a sandbox base image based on the channel or user. It runs outside the sandbox and executes code inside it, using a secret proxy to reach authenticated services [14]. A new PR automatically triggers a review [14].

Shorter notes

- **Compaction (antirez).** Salvatore Sanfilippo disagrees with commenters who say you should never reach context compaction. He says compaction comes up less often for him, and models are less disoriented afterward with Astra and especially Opus 5.5. The model now has an explicit tool to search the session’s JSONL log; before, he had to tell it to look there [15].
- **Spending (antirez).** He had switched his Anthropic plan from 20x to 5x by accident and, using Opus 5.5 heavily, never hit the weekly limit, so he kept it [16]. He argues for flat subscriptions, for example a €200 reimbursement per employee, over paying API rates [16]. He also argues that agents give the most leverage to developers with product vision who steer a project at a high level [16].
- **Ramp-up prompt (TheStandup).** On a new codebase, ask the AI to find the relevant repos, explain why something behaves the way it does, and read the commits to trace the original decision. One host says this covers “at least a good 80%” of the questions that slow onboarding [17].
- **Keep your data portable.** A user said Grok Bot deleted weeks to months of work for him and others [18]. Kent C. Dodds responded by recommending Kody, which keeps your work outside any one walled garden so you can switch agents without a migration [19].

Sources

1. X article by @addyosmani
2. X post by @theo

3. X post by @mitsuhiko
4. X post by @mitsuhiko
5. X post by @robinebers
6. X post by @theo
7. X post by @theo
8. X post by @theo
9. X post by @theo
10. X post by @theo
11. X post by @theo
12. X post by @theo
13. X post by @theo
14. The five components of building a “company harness”
15. Da oggi si banna :) Senza pietà
16. Quanto bisogna spendere in token?
17. Programming Laws Tier list | TheStandup
18. X post by @dbonneville
19. X post by @kentcdodds