// WHITE PAPER · PUBLISHED 27 JULY 2026

False Green

Grading three AI code reviewers across four repositories, and the bug class that reproduced on every stack

Jason Newell · MAX Research Collective · Report WPR-MAX-015 · Published 27 July 2026
Licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0),
https://creativecommons.org/licenses/by/4.0/
Original: https://www.maxresearchcollective.com/articles/false-green.html
Cite as: Newell, J. (2026). False Green. MAX Research Collective, WPR-MAX-015.
https://www.maxresearchcollective.com/articles/false-green.html

A panel on the network console said the firewall was blocking. Behind that word sat a verifier that had opened the firewall's rule file, failed to parse it, and returned green on the way out. Present-but-unreadable rendered the same pixel as present-and-working.

The catch came from a hosted code reviewer about three minutes after the branch went up, on pull request 190. Nobody had looked at it yet.

That shape kept repeating. Over eleven days and four repositories, the thing the machine found most often, and found best, was a status reporting health it had never checked.

Three phases and four repositories

A bake-off is a trial where the candidates work the same job and somebody counts the results. This one ran from July 5 to July 27 in three phases.

Phase one put Gemini Code Assist against Qodo Merge with neither one tuned, and 394 findings got hand-rated across 110 pull requests. Phase two tuned Qodo while Gemini wound down toward its July 17 shutdown. Phase three, which is what the numbers below cover, put CodeRabbit live across the whole fleet under one configuration family.

Four repositories carry that fleet. MAX3 is a local-first AI resident written in Python, and REEF is a Next.js network console. FeedHacker is a TypeScript browser extension. The fourth is the Max Research Collective site, which builds to static files and has no runtime, so the reviewer is switched off there on purpose.

In the eleven-day window the fleet opened 243 pull requests and merged 235. CodeRabbit engaged on 193 of them, which is about 91 percent of the 211 that touched code. Coverage ran 86 percent on MAX3, 96 percent on REEF, and 100 percent on FeedHacker's much smaller volume. The pull requests it skipped were mostly documentation and board updates that the configuration tells it to ignore.

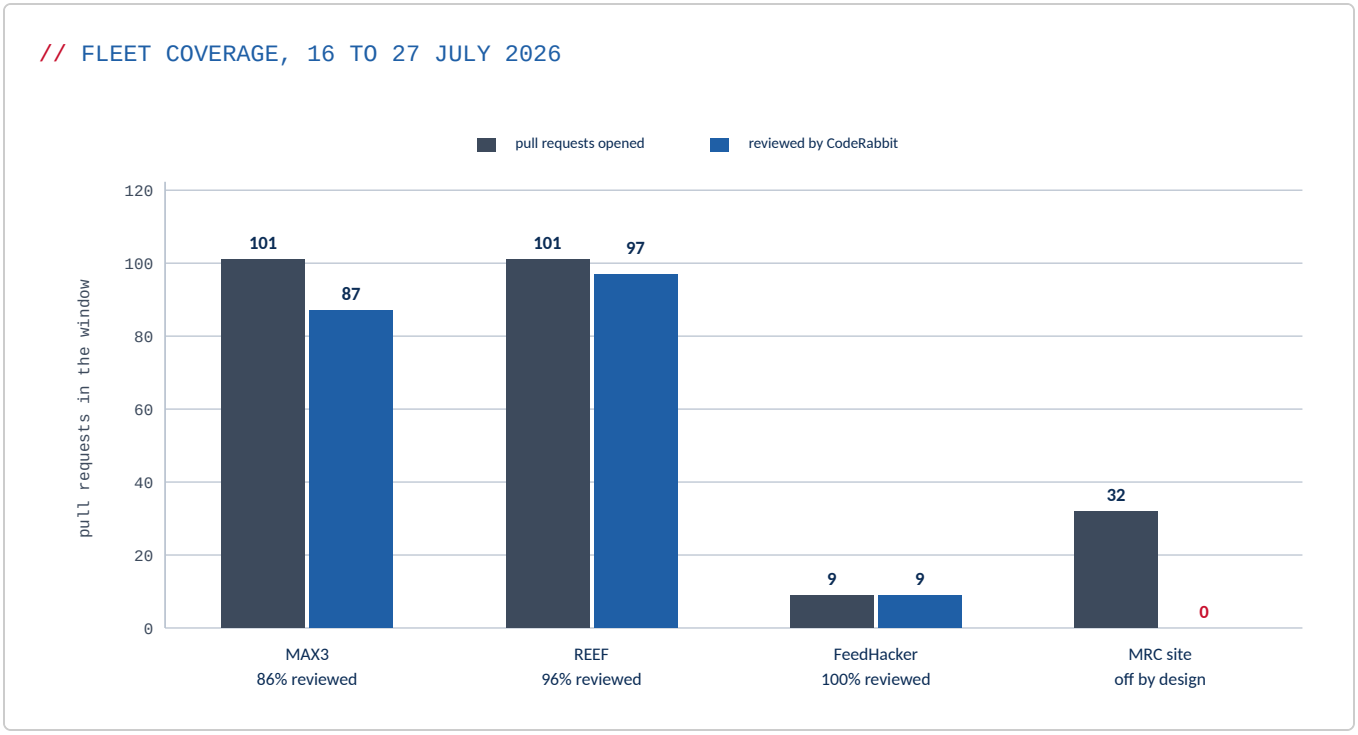


Figure 1. Compare the two bars on MAX3, where the gap is documentation traffic, against the deliberate zero on the site.

One limitation belongs here rather than in a footnote at the end. The three tools never ran as primary reviewer at the same time. Qodo and Gemini owned the earlier span, CodeRabbit took over partway through, and by the end it was working alone. This is a comparison across tenures rather than across identical diffs, and every claim below inherits that.

What the ledger says

REEF keeps a durable issue ledger, and closing a bug there writes a row carrying a “Found by” attribution. There are 189 rows in it. Counted by bug rather than by comment, over comparable spans of roughly a hundred pull requests each, the sole-credit rows land like this.

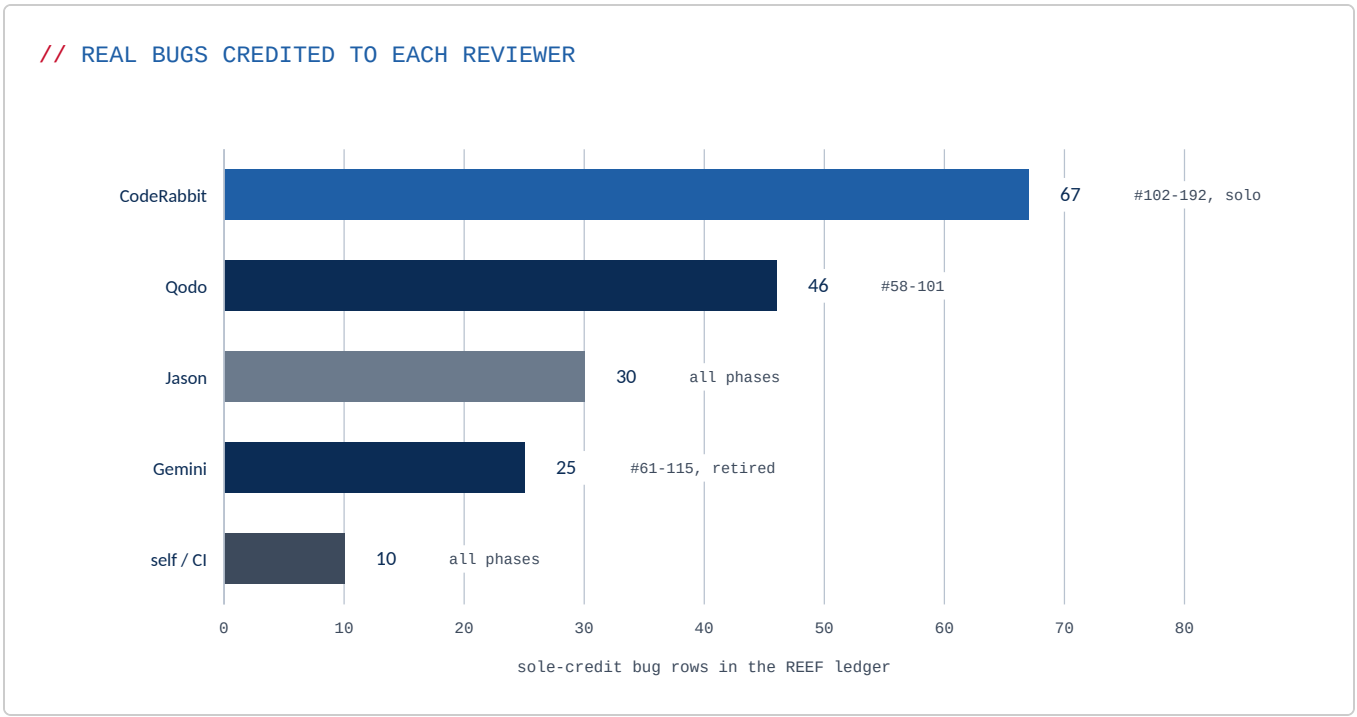


Figure 2. Read the top bar against the two below it, and note that the human row sits fourth.

CodeRabbit holds 67, against Qodo’s 46 and Gemini’s 25. That’s about one and a half times the first and two and a half times the second, produced while running with no second bot beside it.

The human row is 30, across all three phases. A machine found more than twice that in eleven days. The gap is the finding.

The count carries one caution. The ledger writes one row per pull request, so a pull request with ten separate gaps still counts once. Every number in that chart is a floor. They are all floors in the same way, so the ratios survive.

The class it kept finding

The recurring class, the one worth a subscription, is a status reporting that something is up to date when the check underneath it had errored, returned nothing, or gone stale. Call it false green.

On REEF the false-green findings clustered where a wrong answer costs the most. A JSON boolean trap on the fleet-enable endpoint let the string “false” evaluate as true in Python and write a real settings change. A Pi-hole authentication bug turned a 401 into silence, and unknown status defaulted to “blocked,” which is the reassuring answer. A liveness strip held green past staleness. Around 21 of REEF’s 70 CodeRabbit rows land on the security, authentication, or fail-closed surface. None of that is lint.

MAX3 produced its own version of the same thing. One roll-up carried eight instances of a single fail-closed rendering class. A SKOS export shipped as a custom shape while claiming to be JSON-LD. A schema verifier’s if/elif chain short-circuited and under-reported a violation that hit both domain and range. A telemetry path committed on success and lost its retryable state.

FeedHacker runs at a ninth the volume and still gave up a storage migration bug. Removed toggles never reset, so old non-default choices survived silently for existing users. Its structural lint fired a real cross-site scripting flag on an inner-HTML write in the same window.

Then came the moment that settled it. On two repositories the reviewer read its own configuration file and found the error sitting in it. Gemini reviewed that same MAX3 configuration on its last working day. It was wrong on all four points, then conceded on the thread.

Noise fell as the window ran. The first MAX3 roll-up recorded 14 actionable findings against 3 declined, one of which was a true false positive. The second recorded 10 actionable and nothing declined. The third recorded 5 and nothing declined. One hard false positive in eleven days is the whole tally.

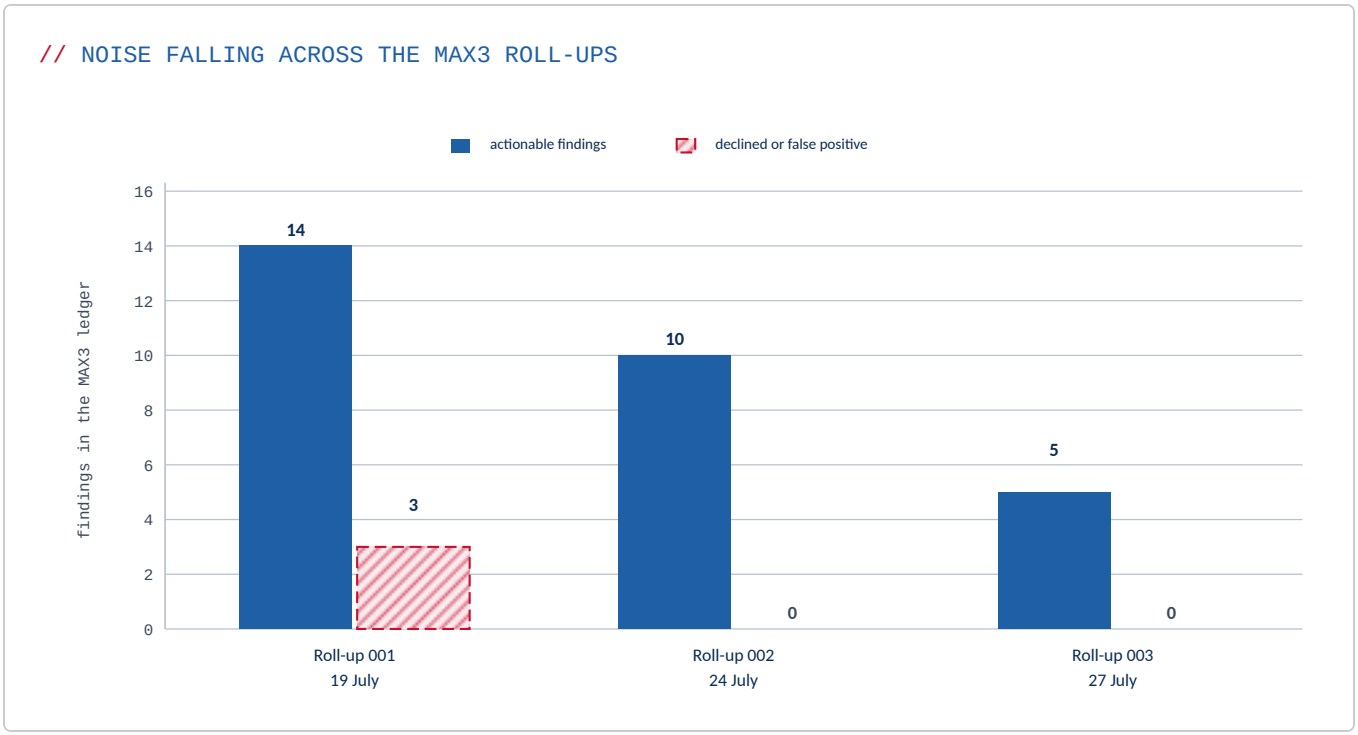


Figure 3. Watch the red column vanish after the first roll-up and stay gone.

Speed, and the hole underneath it

First response landed between two and six minutes on the two high-volume repositories, with a sampled median of 2.7 on MAX3. Gemini's median was 2.2, and it stayed the faster tool right to the end. The continuous-integration gate that blocks a merge sits at ten minutes, so both cleared it with room.

FeedHacker ran slower and much less predictably, from about seven minutes to an hour and 46. Latency there scales with diff size on an asset-heavy repository, which is a property rather than a defect.

One setting turned out to be load-bearing everywhere. The reviewer skips draft pull requests by default, and all three code repositories open their work as drafts. All three hit that blind spot and all three fixed it the same way.

A hole sits underneath those numbers. REEF's median pull request lifetime is 13 minutes, and 55 of 100 merged inside 15. A first pass arriving in three to six minutes still beat most of those merges, which is how coverage stayed at 96 percent. One documented miss: #161 merged on the walkthrough, before its ten findings arrived.

The mitigation is a merge gate that waits fifteen minutes or for the review, whichever comes first.

The flywheel

The output numbers are close enough that a reasonable person could argue them. The learning surfaces are where the argument stops.

Three of them stack. The reviewer reads a standards file kept in the repository, so its findings cite house rules by number instead of offering generic advice. It learns from resolved threads, so a rejected suggestion stays rejected. It also runs a deterministic structural lint called ast-grep, which matches on the shape of code rather than on its text. Nobody else in the trial offered that at all.

Gemini had none of the three. Qodo had the first two.

The third is the one that compounds, and the compounding got measured. REEF's standards file went from 717 lines to 1,945 in eleven days, an increase of 1,228 lines, or roughly 2.7 times its starting size. The sections added in that window came straight out of the reviewer's most-repeated findings: test quality, debounce behavior, honest diagnostics, and the fleet light law.

MAX3 shipped three repository-local ast-grep lints in the same window. Two were written directly from recurring findings, and one of them is named no-synthesized-green-status. It exists to catch the false-green class in the editor before a pull request is ever opened. FeedHacker's smaller standards file grew thirteen new sections.

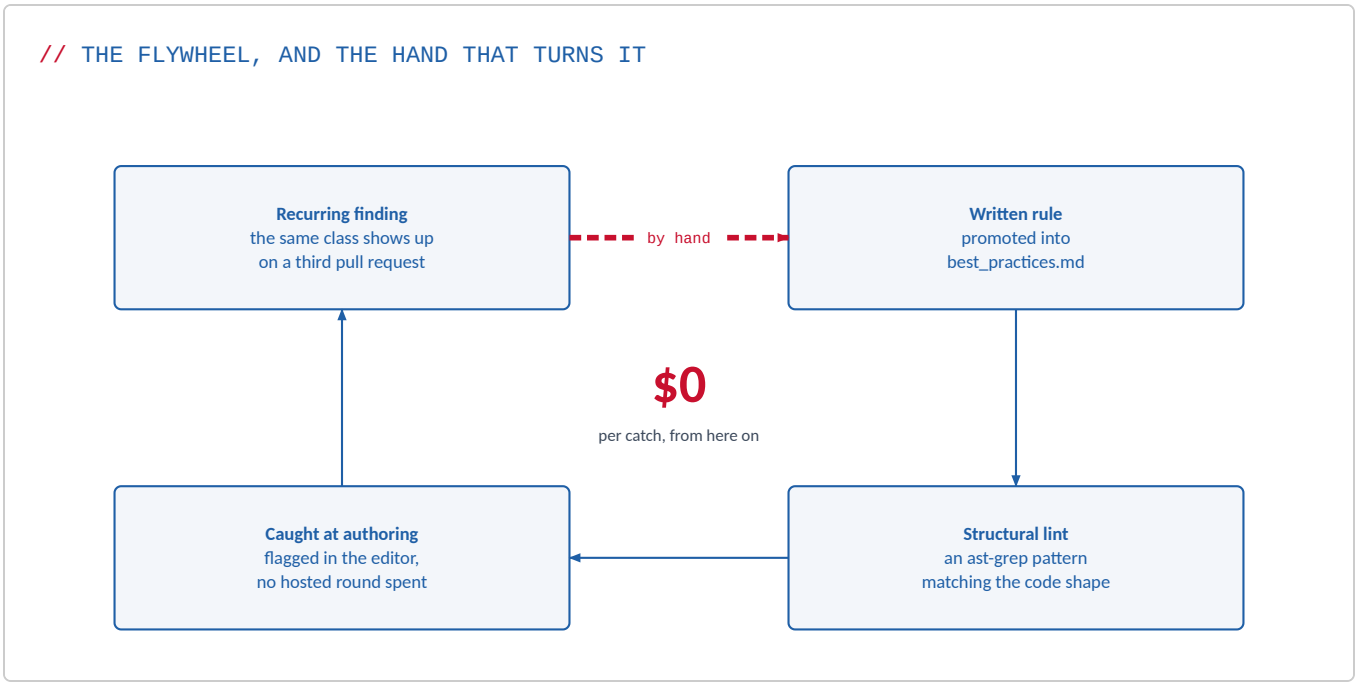


Figure 4. Follow the cycle clockwise, and note that the promotion step at the top is the manual one.

The loop is short. A recurring finding becomes a written rule, the rule becomes a lint, and the lint catches the next instance at authoring time for nothing. The hosted round never gets spent on that class again, so the paid attention moves up to something harder.

None of it happens on its own. The tool finds the class. A person still has to notice it's recurring and then write the rule and the lint. Every lint in the fleet was written by hand after somebody read three findings and saw the same shape three times.

What it costs

The trial ran on a Pro Plus seat with the chill profile, which is the low-noise setting. Standard Pro carries unlimited reviews at roughly $24 to $30 a month for one seat. Normalized against measured volume, that works out to about $0.09 to $0.11 per reviewed code pull request.

The seat is licensed per account rather than per repository. One login covers all four repositories at once. None of the three candidates consumed continuous-integration minutes either, so the choice of reviewer adds nothing to the separate Actions bill.

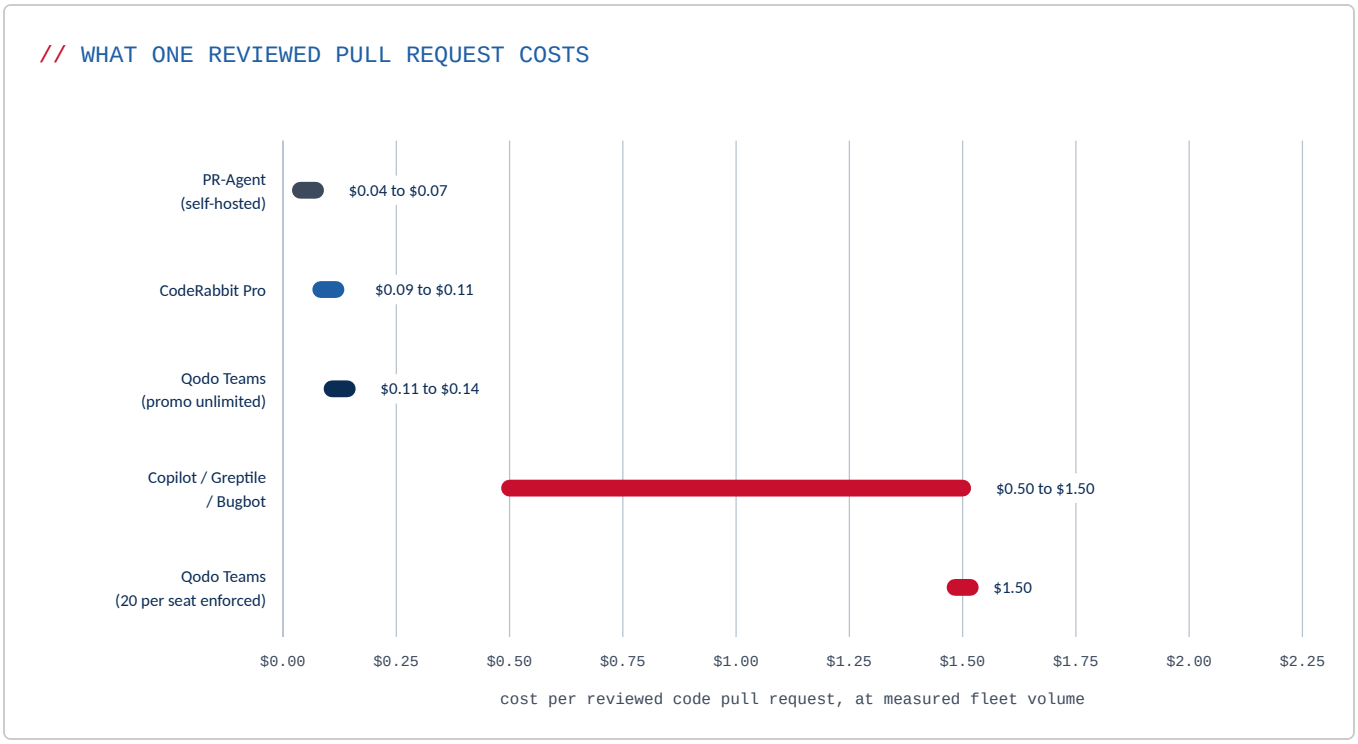


Figure 5. Note the distance between the flat-rate band on the left and the per-review options on the right.

Qodo's number looks similar until the fine print. Its unlimited tier is a time-limited promotion sitting on top of a documented standard of 20 pull requests per seat per month. At 450 to 600 organization pull requests a month, honoring that standard means 23 to 30 seats. The vendor quote came back near $2,040 a month. That is roughly seventy times the CodeRabbit line for the same work.

The per-review-priced tools were rejected on arithmetic alone. Copilot code review lands around $130 to $190 a month at this volume, Greptile around $260 to $330, and Cursor Bugbot between $280 and $525. Ten to twenty times flat rate to run an autonomous loop.

The cheapest option in the field isn't hosted at all. PR-Agent, self-hosted and running on our own key, works out to $0.04 to $0.07 per pull request. It costs continuous-integration minutes and maintenance instead of a subscription. It stays in the repositories as break-glass, because an Apache-licensed reviewer running on our own key can't be sunset by a vendor.

The bill that arrives in attention

The one measured drag came from MAX3, and the cause was on my side of the keyboard.

Rate limiting hit about six times in the first roll-up. The second produced 13-minute waits across roughly 15 reviews on 9 pull requests. The third hit twice on a single pull request. Every incremental commit re-triggers a full review, and a push that lands mid-review causes a restart.

No finding was ever lost to it, and neither REEF nor FeedHacker surfaced the limit in the samples. That makes it a measurement of a commit habit rather than a tool defect. Pushing five small fixes to an open pull request while a reviewer is working means paying for five reviews and reading one.

The fix is unglamorous. Run the local pass before opening the pull request. Ship standards-correct code with the full test triad in the first commit. Batch the fix commits, and leave the branch alone while a review is running.

The repository where it stays off

The Max Research Collective site opened 32 pull requests in the window. It received zero reviews and zero findings, plus a single draft skip. Its configuration disables the reviewer outright, committed on July 20.

That is the intended result. A static site with no build-time logic has no runtime failure paths for a reviewer to reason about, and a reviewer that finds nothing still costs attention to read.

The caveat is real. A minority of those pull requests do touch logic, including an in-browser retrieval script, a Python build step, and workflows carrying SFTP and SMTP secrets. If that surface grows, selectively re-enabling review there is worth revisiting.

A sharper trap sits in the same corner. The seat is licensed to a single user. A pull request opened under any other login goes silently unreviewed, with no warning anywhere in the interface. Every window pull request across all four repositories came from the licensed account, which is the only reason the coverage numbers mean anything.

The call

Keep CodeRabbit Pro on the three code repositories at one seat, and keep it off on the site. Gemini is already gone. The one thing it owned that the others didn't was documentation coverage, and a drift lint handles that at no cost.

Drop paid Qodo as a second lane. The tuned configuration and its standards file stay in the repositories as break-glass assets, unsubscribed.

Keep the on-box ModelCouncil as the free second voice. The most useful structural finding of the whole bake-off was that two reviewers with near-disjoint output, overlapping around two percent, are purely additive. That one runs on our own keys, with no seat and no sunset risk.

Two lanes is the cap. The tail of low-severity findings gets paid for in attention, and attention is the scarce resource in a shop of one.

METRIC	GEMINI CODE ASSIST	QODO MERGE (TUNED)	CODERABBIT (TUNED, FULL WINDOW)
Real bugs credited	25 sole-credit rows	46 sole-credit rows	67 sole-credit rows, produced while running alone
Signal	about 80 percent actionable, 19 at severity one	27 percent out of the box, near 100 percent tuned	about 39 actionable as a floor, zero noise on two of three roll-ups, one true false positive
Speed, first response	2.2 minutes, the fastest of the three	about 5.2 minutes	two to six minutes, median about 2.7 on MAX3
Coverage	100 percent including documentation, blind to drafts	code pull requests, documentation skipped	about 91 percent of code-repository pull requests, 193 of 211, drafts reviewed
Learning	none	reads a standards file, learns from accepted suggestions	standards file, resolved threads, and ast-grep structural lints
Cost	free, then ceased on 17 July	about $30 a month if unlimited holds, carries a promotion cliff	$24 to $30 a month on the standard tier, one seat covering four repositories
Availability	ceased	continues, with pricing risk	continues, unlimited on the standard tier

Table 1. Read the learning row and the availability row together, which is where the three separate.

What this doesn’t establish

The MAX3 actionable count, about 39, is a documented floor rather than a census. A real number requires pulling reviews on all 87 engaged pull requests, and that hasn't been done.

Per-pull-request latency and findings-per-pull-request are sampled at three to eight pull requests per repository. They aren't aggregated fleet-wide.

No clean same-code three-way ran in phase three, because Gemini was dead and Qodo was winding down. Phase-three quality is judged against the earlier baselines and against the two cross-repository replications. That is weaker evidence than an identical-diff head-to-head would have been.

MAX3's failure-pattern file carries no "Found by" tagging, so MAX3 can't be counted the way REEF and FeedHacker were. Its number comes from the roll-ups instead.

The seven-metric scoring is a working zero-to-five rubric from one rater, derived from the sections above. It organizes a judgment and it doesn't replace one.

The review footers show a trial tier. The standard monthly price, and what the higher tier adds over it, has to be reconfirmed at signup. The whole recommendation assumes unlimited stays on the standard plan, which is the exact assumption that broke on Qodo.

The verifier on REEF fails closed now. When it can't parse the firewall file, the panel goes amber and names the file it couldn't read. That took one comment on a pull request that stayed open eleven minutes, and a rule in a standards file that will still be there when the vendor isn't.

Appendix A. Glossary

Terms are listed alphabetically, in the sense they carry in this evaluation.

ast-grep

A linter that matches on the parsed structure of code rather than on its text, so a rule can describe a shape instead of a string. It is the surface no other reviewer in the trial offered, and it turns a recurring review finding into a free catch in the editor.

best_practices.md

A standards file kept inside a repository, holding the house rules a reviewer is expected to enforce. A reviewer that reads it cites rule numbers instead of offering generic advice.

break-glass

An asset kept configured and unused, ready if the primary fails. The tuned Qodo configuration stays in the repositories on that footing after the subscription ends.

continuous-integration gate

An automated check that blocks a merge until it passes. This fleet sets its hard gate at ten minutes, which is the deadline a hosted reviewer has to beat.

cross-site scripting

A class of web flaw where attacker-supplied text is written into a page as markup and then executed. Writing untrusted content through an element’s inner HTML is the common route in.

draft pull request

A pull request marked as work in progress. Most hosted reviewers skip drafts by default, which blinds them to any workflow that opens work in that state.

fail-closed

Behaviour where a component that cannot confirm a safe condition reports failure rather than success. The opposite habit, reporting success when the check never completed, is what this evaluation kept finding.

false green

A status or diagnostic that reports health it never verified, because the check underneath it errored, returned nothing, or had gone stale. It is the highest-value finding class in this window and it reproduced on all three code stacks.

false positive

A finding that names a defect where none exists. One hard false positive turned up across the eleven-day window.

incremental review

A re-review triggered automatically when new commits land on an open pull request. It keeps a long-lived branch covered and it also multiplies the review cost of a branch that is pushed to repeatedly.

JSON-LD

A standard way of expressing linked data in JSON so that machines can read the relationships, not only the values. An export that mimics its look without following the standard will parse and still mean nothing.

lint

An automated check that flags code matching a known bad pattern, run at authoring time rather than at review time. A lint costs nothing per run once it exists.

ModelCouncil

The on-box second reviewer in this fleet, running on local keys rather than a subscription. It carries no seat, no quota, and no sunset risk.

negative control

A case included in a trial precisely because the treatment should do nothing there. The static site plays that role, and its zero findings confirm the configuration rather than undermining it.

pull request

A proposed change to a repository, opened for review before it merges. It is the unit everything in this evaluation is counted against.

rate limiting

A vendor cap on how many reviews an account can trigger in a period. Here it appeared as waiting time on one repository, caused by pushing commits into an

open review.

shift-left

Moving a check earlier, from review time back to authoring time. Every lint promoted out of a finding is one instance of it.

SKOS

A vocabulary standard for publishing thesauri and taxonomies as linked data. An export claiming to be SKOS has to follow the shape, or downstream tools read nothing.

sole-credit row

A ledger entry crediting exactly one finder with a bug. It is the count used here because it cannot double-count a defect two reviewers both noticed.

test triad

The unit, integration, and system tests a change is expected to carry before it opens. Shipping all three in the first commit is what keeps a review to a single round.

Appendix B. Figure and table data

Each figure is drawn from the datasets below, so the document stands on its own without the underlying repositories.

Figure 1, sole-credit bug rows

Source: the REEF issue ledger, counting one row per bug, over comparable spans of roughly one hundred pull requests.

FINDER	SOLE-CREDIT ROWS	SPAN
CodeRabbit	67	#102 to #192, this window, running alone
Qodo	46	#58 to #101, before the window
Jason	30	all three phases
Gemini	25	#61 to #115, retired 17 July
self or CI	10	all three phases

Figure 2, fleet coverage

Source: enumeration of pull requests opened between 16 and 27 July 2026, matched against reviewer engagement.

REPOSITORY	OPENED	MERGED	REVIEWED	COVERAGE
MAX3	101	97	87	86 percent
REEF	101	101	97	96 percent
FeedHacker	9	7	9	100 percent
MRC site	32	30	0	disabled by design
Fleet total	243	235	193	about 91 percent of 211 code pull requests

Figure 3, MAX3 signal ratio

Source: the MAX3 findings ledger roll-ups, all of them CodeRabbit.

ROLL-UP	ACTIONABLE	DECLINED OR FALSE POSITIVE
001, 19 July	14	3, one of them a true false positive
002, 24 July	10	0
003, 27 July	5	0

A separate pull request, #1314 on 18 July, carried ten actionable findings across two rounds and sits outside the roll-ups. The documented total of about 39 is therefore a floor rather than a census.

Figure 4, the flywheel

Layout rather than a numeric series. A recurring finding is promoted to a rule in the standards file, the rule is expressed as an ast-grep pattern, and the pattern catches the next instance at authoring time for nothing. Measured instances in the window: the REEF standards file grew from 717 lines to 1,945, and MAX3 shipped three repository-local lints, two of them written straight from recurring findings.

Figure 5, cost per reviewed pull request

Source: normalized against measured fleet volume for the window.

OPTION	COST PER REVIEWED CODE PULL REQUEST
PR-Agent, self-hosted	$0.04 to $0.07
CodeRabbit Pro	$0.09 to $0.11
Qodo Teams, promotional unlimited	$0.11 to $0.14
Copilot, Greptile, or Bugbot	$0.50 to $1.50
Qodo Teams, 20 per seat enforced	about $1.50

Table 1, the seven-metric scoring behind the comparison

A working zero to five rubric from one rater, derived from the sections above. Zero is absent or worst and five is best. It organizes a judgment rather than replacing one.

AXIS	GEMINI	QODO (TUNED)	CODERABBIT
Findings and output	3	4	5
Speed	5	3	4
Efficiency	4	4	4
Learning	0	3	5
Cost	5	2	4
Quality	3	4	4
Availability	0	3	5

MAX3 WPR-MAX-015

// ABOUT THE AUTHOR

Jason Newell leads organizations and people in technology, and is a researcher, deep technologist and builder — from a CRAY to agentic AI. He founded MAX Research Collective to build AI for regulated industries that runs where its owner has control, with healthcare as its strongest focus. About Jason → Connect on LinkedIn →

Newell, J. (2026). MAX Research Collective, WPR-MAX-015. CC BY 4.0