

ZERO-TRUST HARNESSES FOR AGENTIC SOFTWARE DELIVERY

OHANS EMMANUEL

Co-founder & CEO, Coldtea.ai

I've been coding with agents for months

99%

of my lines in September were AI-written



■ AI lines 213.1K ■ Human lines 2.6K

TOTAL TIME

280h 6m

DAILY AVERAGE

9h 20m

avg user 1h 55m

TOKENS

12.5B

12.3B cached

PROMPTS

7K

1.4K avg length

LLMs fundamentally cannot write great software

even though they do a good job of solving localised problems
and can mimic the distribution of programming



Scan to answer

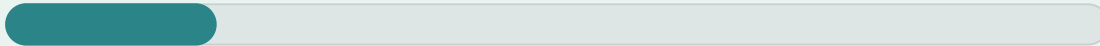
Agree or not?

Do you agree?

Yes 49% 61



No 19% 24



Not sure 32% 40



125 VOTES



Scan here

"I love how most of the sw talks nowadays are actually QA talks"

"I live AI, but it doesn't love me back"

"Definitely needs a human in the driver's seat"

"toimitusjohtaja toimitusjohtaja" "no" "<3" ":3"

"You need newest tools" "is this chat censored" "depends model and harnessa"

"For sure! Easy to agree to! Then again... Not many individuals, not many teams can 😊"

"Needs the human to innovate still" "Creativity should be the source of great software"

"We need to plan and design and guide then it can be good"

"The customer owns the definition of valuable."

"case one: happy accidents :)"

"Try asking AI to build, look away and review the code after a few days :)"

"LLM is just one component of whatever "AI" is tomorrow."

"Well guided and properly isolated, good enough"

"Different expectations of greatness"

"set of hooks for each coding agent lifecycle step"

"Essentially it depends on the code they have learned, not fundamentally"

"SW is deterministic, AI is not"

"Been trained with the less optimal SW humans have built"

"It needs nec3ssary measures"

"They never write smart code, only average."

"Need someone to direct and validate"

"Processes are needed"

"Over abstraction, adds additional related features and more"

"We are not coding for ai but for people"

"Hei Äiti!" "Definition of great?" "it's all taste"

"Just wait"

"Not great for complex or refined, but great for fast and narrow"

"We need a world model."

"With correct guidelines and rails it can do pretty well"

"What is great software?"

"Let's where this is going"

"Define good code."

"Average by design"

"Depends on what "good" is defined as"

"It needs guardrails because otherwise it doesn't have a goal."

"Too early to say"

"Taste is still fundamentally human."

"They can't on their own, but could with human direction"

"Might be a skill issue"

"They are writing "average" code. Statistically"

"By what definon of "great software"?"

"--"

"You need to build the environment for them"

"Human vision and direction"

"They just can't stop writing 😊"

"We need people for guidance"

"Depends on how you define it."

"N/a"

What the room says



Add yours

56 MESSAGES

Two sides



Hi, I'm

Ohans Emmanuel

Cofounder & CEO, **Coldtea**

www.coldtea.ai



The IDE for the agentic SDLC.
Plan. Build. Verify. Ship. Watch. Repeat.

PREVIOUSLY

Revolut



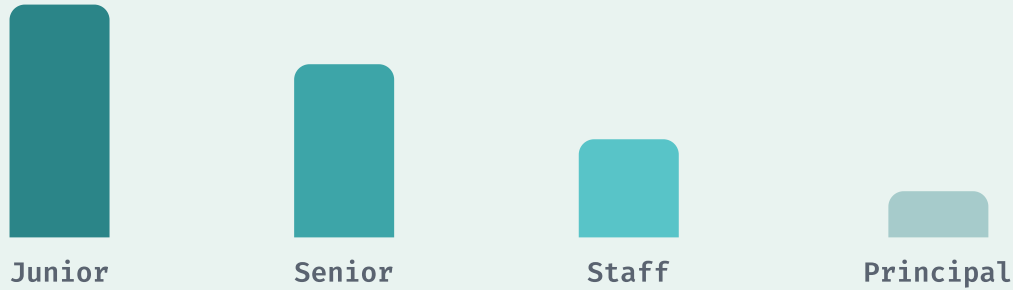
omio

Zero-trust harnesses

Principles for verifying each stage of the software development lifecycle. Don't trust, verify.

Building was never the hard part

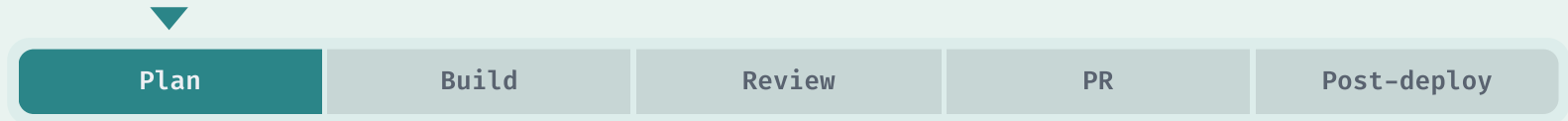
HOW MUCH CODE YOU WRITE



At any real scale, the job is **everything around the code**, not simply typing.

Checkpoint 01

Plan



Write it down somewhere every later step can **point back to**. Not a file on your laptop.

Checkpoint 02

Build



Plan

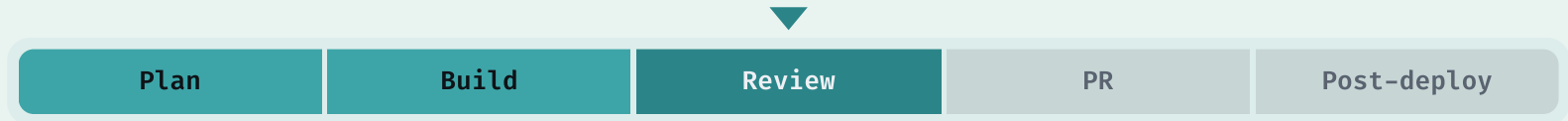
Build

Review

PR

Post-deploy

Review



We review **implementation plans**, not pull requests

We run **two models** against each other until there are no comments left

We review **visually** to keep a sharp mental model of a system that changes daily

We review **implementation plans**, not pull requests

PR-comment Redis mutex deletes a lock it may no longer own, reopening the duplicate-table race

Problem Statement

The mutex added to serialize per-PR QA comment upserts can release a lock it no longer holds. It acquires with `SET key "1" NX PX 30000` and releases with an unconditional `redis.del(key)`, using a fixed value with no ownership token. The guarded work paginates every comment on the PR, then creates or updates a comment and deletes duplicates, which on a busy PR can run past the 30-second TTL. When it does, the lock expires, a second run acquires it, and the first run's `del` deletes the second run's lock. A third run can then acquire while the second is still inside the critical section, so two syncs execute the non-atomic find-then-create at once, which is the duplicate QA-table race this change set out to prevent.

Solution

Make the release safe against expiry by storing a unique token as the lock value and deleting only if the stored value still matches, using a compare-and-delete (a small Lua script or `GET -then-conditional- DEL`). That way a run whose work outlived the TTL cannot delete a lock another run now holds, and the mutex keeps its guarantee.

Evidence

```
withMatchaPrCommentLock ( MatchaQa.GitHubClient.ts:809 ) sets the lock at :824 with nx: true, px: MATCHA_PR_COMMENT_LOCK_TTL_MS (30_000) and a constant value "1". The finally block at :838 calls redis.del(key) unconditionally. With no fencing token the delete is not tied to ownership. The guarded run() performs an octokit.paginate over all issue comments plus a create/update plus N duplicate deletes, which is unbounded in PR comment count and subject to GitHub rate limits, so exceeding 30 seconds is realistic on active PRs.
```

Classification

Introduced by 51b8dde8 ("serialize PR-comment upsert to stop duplicate QA tables"). The same commit adds a dedupe backstop that keeps the earliest marker-matching comment and deletes the rest on the next sync, so the failure self-heals to a single table rather than permanently duplicating. That bounds the blast radius but does not restore mutual exclusion, so this is a real gap in a newly introduced concurrency primitive rather than a hard regression. A secondary, documented fail-open path (after 25 attempts a loser runs `run()` without the lock while the winner's TTL is still live) leans on the same backstop.

Subtasks

+ Add subtask

Properties

- In Progress
- High
- Assignee

Labels

- Maintenance

Agent artifacts

- Implementation Plan (v1)
- Session logs 1
- PR #249

Cloud runs

- Claude Completed 22 Jul at 9:10 phansemmanual/coldtea PR #249 View session

Get to work

Agent: Claude

Project: coldtea

Local Cloud

Start working on COL-569

Open in workspace

Checkpoint 03: Review

We run **two models** against each other until there are no comments left

The screenshot shows a code editor interface with a sidebar on the left containing 'Home', 'Tasks', 'Terminal', and 'Agentic Testing'. The main editor area is titled 'PAIR (REVIEW) - CHECKOUT' and shows a code file 'Adding Promo Code Discounts' with the following code:

```
66 + items: [
67 +   { sku: "tea-01", kind: "physical", priceCents: 1200,
68 +     qty: 1 },
69 +   { sku: "gift-25", kind: "gift_card", priceCents: 2500
70 +     qty: 1 },
71 + ],
72 + // 10% off the $12.00 tea only: $37.00 - $1.20
73 + assert.equal(applyDiscount(cart, "TEA10"), 3580);
74 +
75 + test("a gift-card-only cart gets no discount", () => {
76 +   const cart = { items: [{ sku: "gift-25", kind: "gift_card",
77 +     priceCents: 2500, qty: 1 } ] };
78 +   assert.equal(applyDiscount(cart, "TEA25"), 2500);
79 +   assert.equal(applyDiscount(cart, "WELCOME"), 2500);
80 + });
81 + test("percentage discounts round to whole cents", () => {
82 +   const cart = { items: [{ sku: "tea-01", kind: "physical",
83 +     priceCents: 1999, qty: 1 } ] };
84 +   // 10% of $19.99 is $1.999 -> rounds to $2.00
85 +   assert.equal(applyDiscount(cart, "TEA10"), 1799);
86 +   assert.equal(Number.isInteger(applyDiscount(cart, "TEA10")), true);
87 + });
88 + let failed = 0;
89 + for (const [name, fn] of tests) {
90 +   try {
```

Below the code, the terminal output shows:

```
● Bash(npm test)
  > checkout@0.2.0 test
  > node test/pricing.test.mjs
  ... +15 lines (ctrl+o to expand)

● All 12 tests pass. Let me hand off to the reviewer.

● Calling coldtea 2 times... (ctrl+o to expand)

* Vibing... (45s · ↓ 3.2k tokens)
```

The right sidebar shows a 'REVIEWER' panel with the following content:

```
model: gpt-5.6-sol medium /model to change
directory: ~/Documents/dev/checkout
permissions: YOLO mode
```

Tip: Start a fresh idea with /new; the previous session stays in history.

You have 3 usage limit resets available. Run /usage to use one.

author: applyDiscount(cart, code) is implemented in src/pricing.mjs; 12 tests pass via npm test.

Design: PROMO_CODES table maps TEA10/TEA25 to {kind:"percent"} and WELCOME to {kind:"flat", value:500}. applyDiscount computes subtotal(cart), returns it unchanged for an unrecognized code, otherwise subtracts a discount computed against discountableCents(cart) – a new private helper that sums non-gift-card line items.

Where to look first:

1. The zero-floor. There is no explicit Math.max(0, ...) – clamping is implicit because discount <= discountable <= subtotal. Percent discounts are a fraction of discountable; the flat WELCOME is Math.min(500, discountable). I think this holds for all non-negative inputs, but it's the invariant most worth attacking.
2. Rounding. Math.round on discountable * value, applied once to the whole discountable amount rather than per line item. Covered by "percentage discounts round to whole cents" (10% of 1999 -> 200, total 1799).
3. Gift cards. Covered by two tests: mixed cart (TEA10 on tea+gift card -> 3580) and gift-card-only (no discount, returns face value).

One spec gap I did not build: TASK.md says "unknown or expired code is a no-op", but no code in the table has expiry data and the cart carries no date, so there is nothing to expire against. Unknown codes are a no-op; expiry is unimplemented. Flag if you think that needs a stub now rather than when Finance supplies dates.

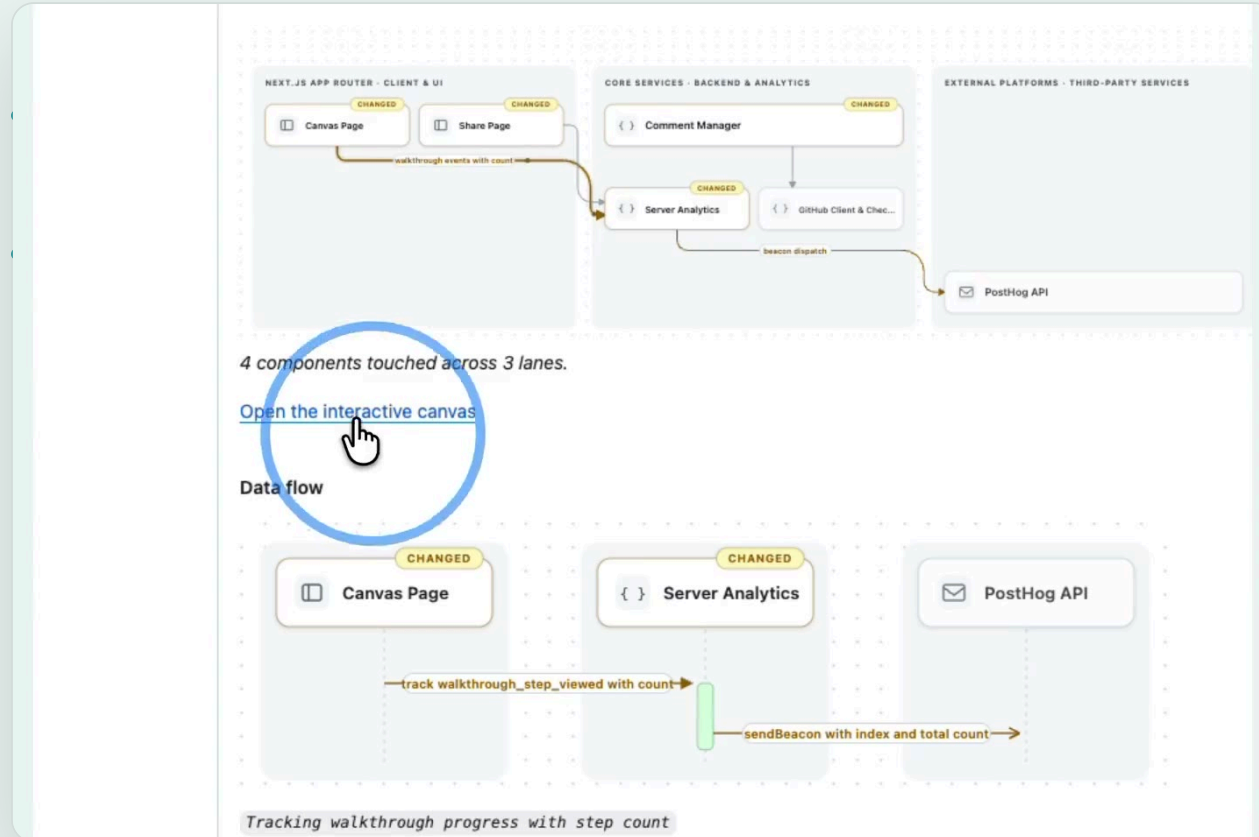
New tests are in test/pricing.test.mjs above the runner block.

Working (0s · esc to interrupt)

Checkpoint 03: Review

We review **visually** to keep a sharp mental model of a system that changes daily

- Code review's value, beyond catching bugs, is maintaining a **mental model** of a software



keeping the

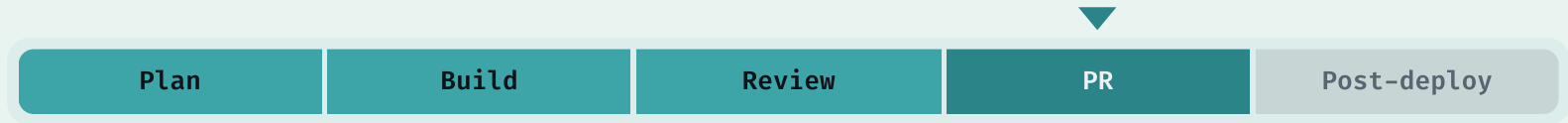
he s



Open source core

Checkpoint 04

PR



We run **automated regression testing** via visual QA agents

We add **exploratory** visual QA review on every PR

We run automated regression testing via visual QA agents

The screenshot displays a GitHub pull request for the feature "feat: apply promo codes to the whole cart subtotal #6". The pull request is open, and the Vercel bot has commented with a deployment to Preview. Below this, the Matcha QA bot has commented with the results of an end-to-end test run. The test results show that one test failed and three tests passed. The failed test is "Gift cards are not discounted", which failed with a duration of 1m 1s. The other three tests, "Add Jasmine", "Promo code TEA10 takes 10% off", and "Guest checkout completes", all passed with durations of 1m 1s, 53s, and 46s respectively.

feat: apply promo codes to the whole cart subtotal #6
maya wants to merge 4 commits into master from feat/discount-applies-to-subtotal

vercel Bot commented 29 minutes ago · edited

The latest updates on your projects. Learn more about [Vercel for GitHub](#).

Project	Deployment	Actions	Updated (UTC)
checkout	● Ready	Preview , Comment	Jul 15, 2026 10:22pm

Review with Vercel Agent

vercel Bot deployed to Preview 29 minutes ago [View deployment](#)

coldtea Bot commented 29 minutes ago · edited

❌ Matcha QA — 1 failed · 3 passed

End-to-end checks against the [preview deployment](#) for checkout.

Test	Result	Duration
Gift cards are not discounted	❌ Failed	1m 1s
Add Jasmine	✅ Passed	1m 1s
Promo code TEA10 takes 10% off	✅ Passed	53s
Guest checkout completes	✅ Passed	46s

► Details

Development

Successfully merging this pull request may close these issues.

None yet

Notifications [Customize](#)

[Unsubscribe](#)

You're receiving notifications because you authored the thread.

1 participant

[Lock conversation](#)

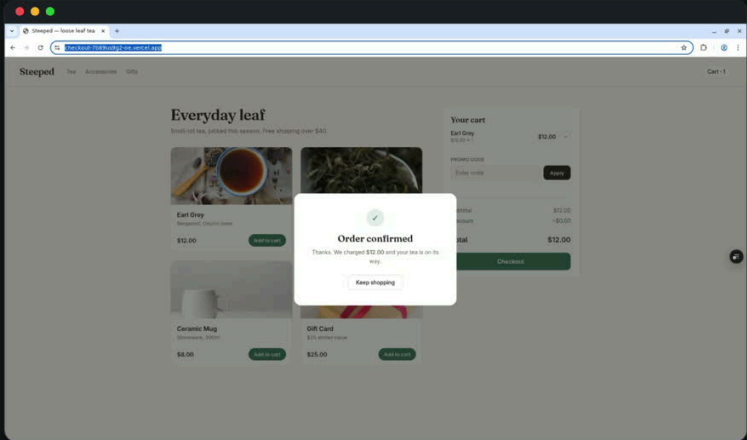
We add **exploratory** visual QA review on every PR

Bluebag / checkout

Guest checkout completes

The QA run successfully completed the scenario: Earl Grey was added to the cart, checkout was clicked, and the order confirmation appeared showing the correct \$12.00 total. **PASS**

- ✓ Add Earl Grey to cart GO TO 0:11
- ✓ Click Checkout GO TO 0:21
- ✓ Verify order confirmation and total GO TO 0:26
- 🗎 Matcha said
The order confirmation message 'Order confirmed' is visible on the page, and it states that the charged total is \$12.00, which matches the user's goal perfectly.

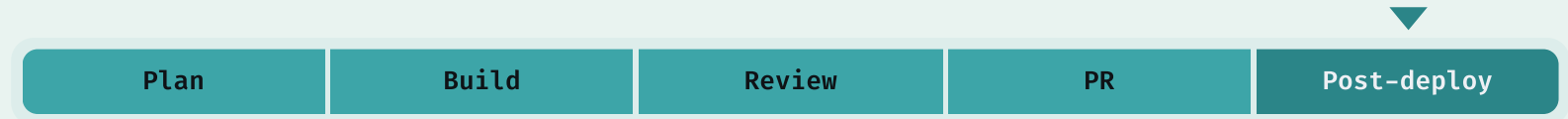


How do you rate this test?

Helpful

Not helpful

Post-deployment

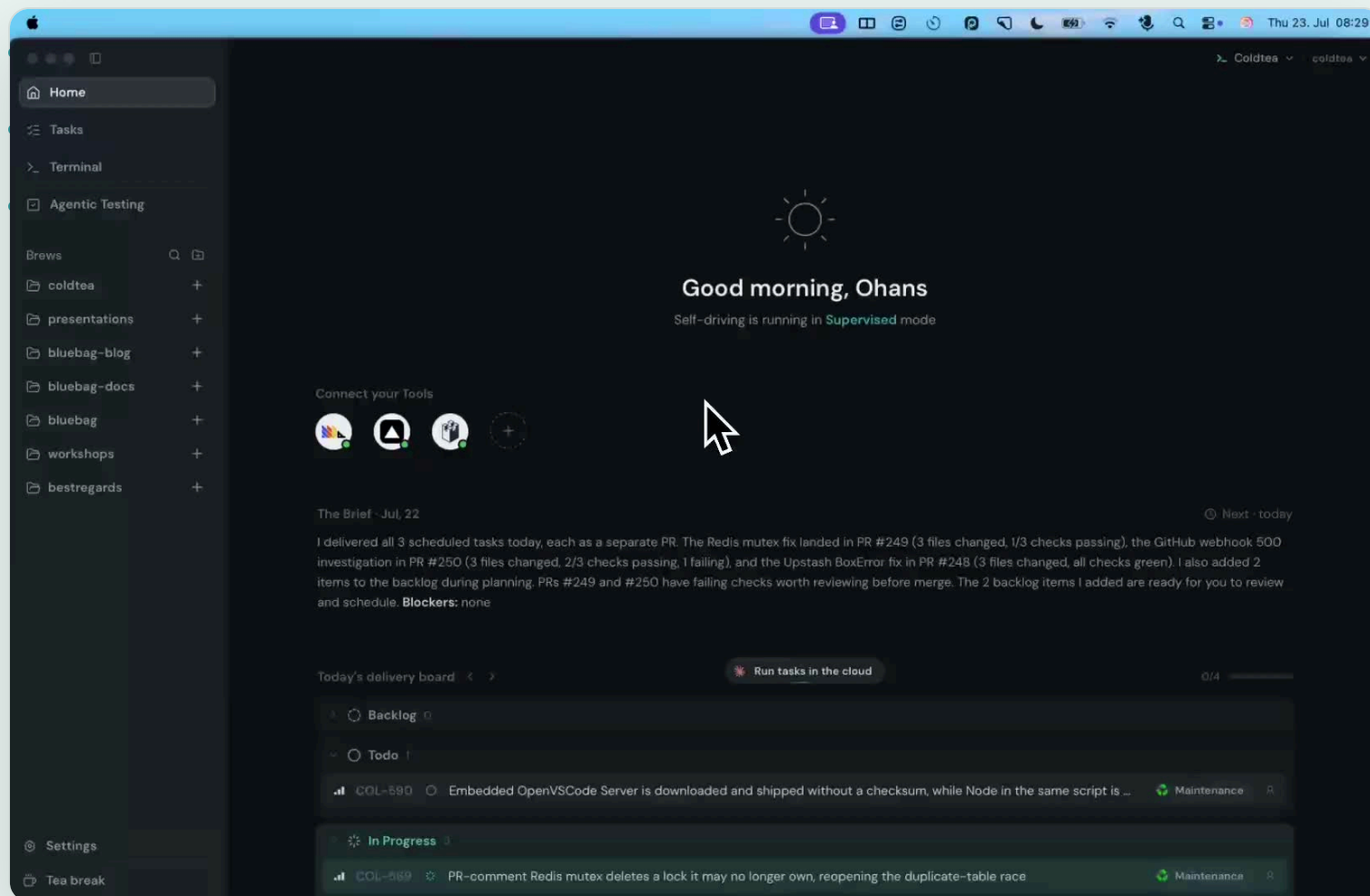


It's live. And it will still break. That's software.

Background agents watch every signal and create tasks with [investigated fixes](#)

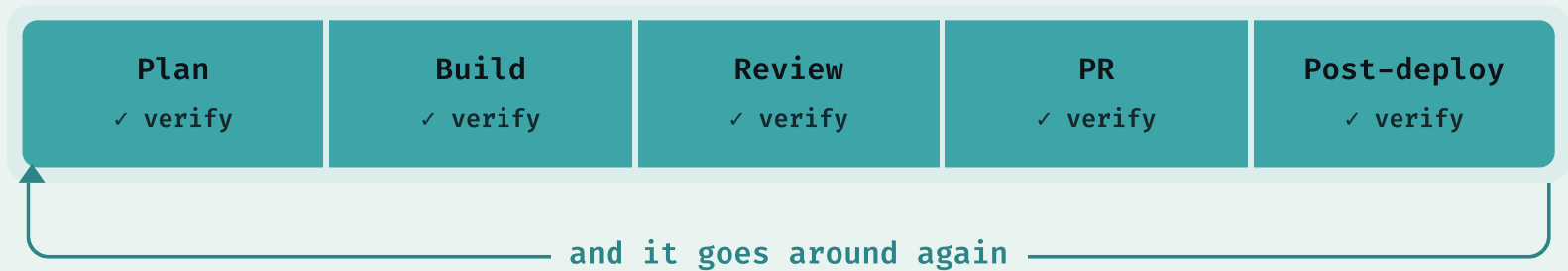
Checkpoint 05: Post-deployment

Background agents watch every signal and create tasks with investigated fixes



It's the combination

Zero trust at **every** checkpoint, or it isn't zero trust.



Coldtea.ai

Ship at agent speed. Break nothing.



Questions? @ohansemmanuel