

A CTO'S GUIDE TO GATING AGENT-WRITTEN CODE

Who checks the agents?

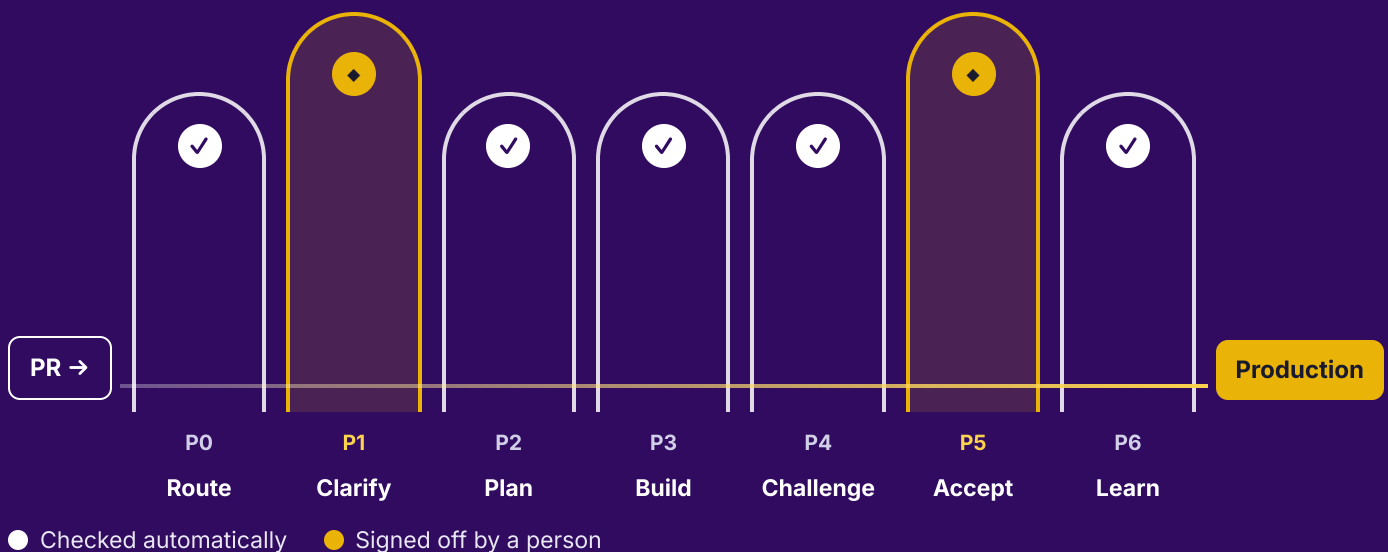
When one model writes the code, the tests, and the review, every check agrees with itself. The seven gates a CTO should put between an agent and production.

**ONE MODEL, EVERY CHECK****Writes, tests, reviews, approves**

0 independent checks

**INDEPENDENT GATES****Every change earns its way to production**

7 gates · 2 human sign-offs



Our delivery process was designed around human mistakes.

Peer review, standups, and release checklists grew up to catch what people get wrong: missed steps, tired eyes, and attachment to their own code. Agents rarely make those mistakes. They make different ones, and they make them with full confidence, at machine speed, across every change. If your gates never look for agent mistakes, those mistakes reach production.

HUMAN FAILURE MODES

Forgetting

Fatigue

Ego

Skipped steps

Context switching

Answered by peer review, checklists, and process.

MODEL FAILURE MODES

Stops too early

Agrees with the prompt

Drifts over long sessions

Invents with certainty

Games the check

Need gates that produce evidence, not opinions.

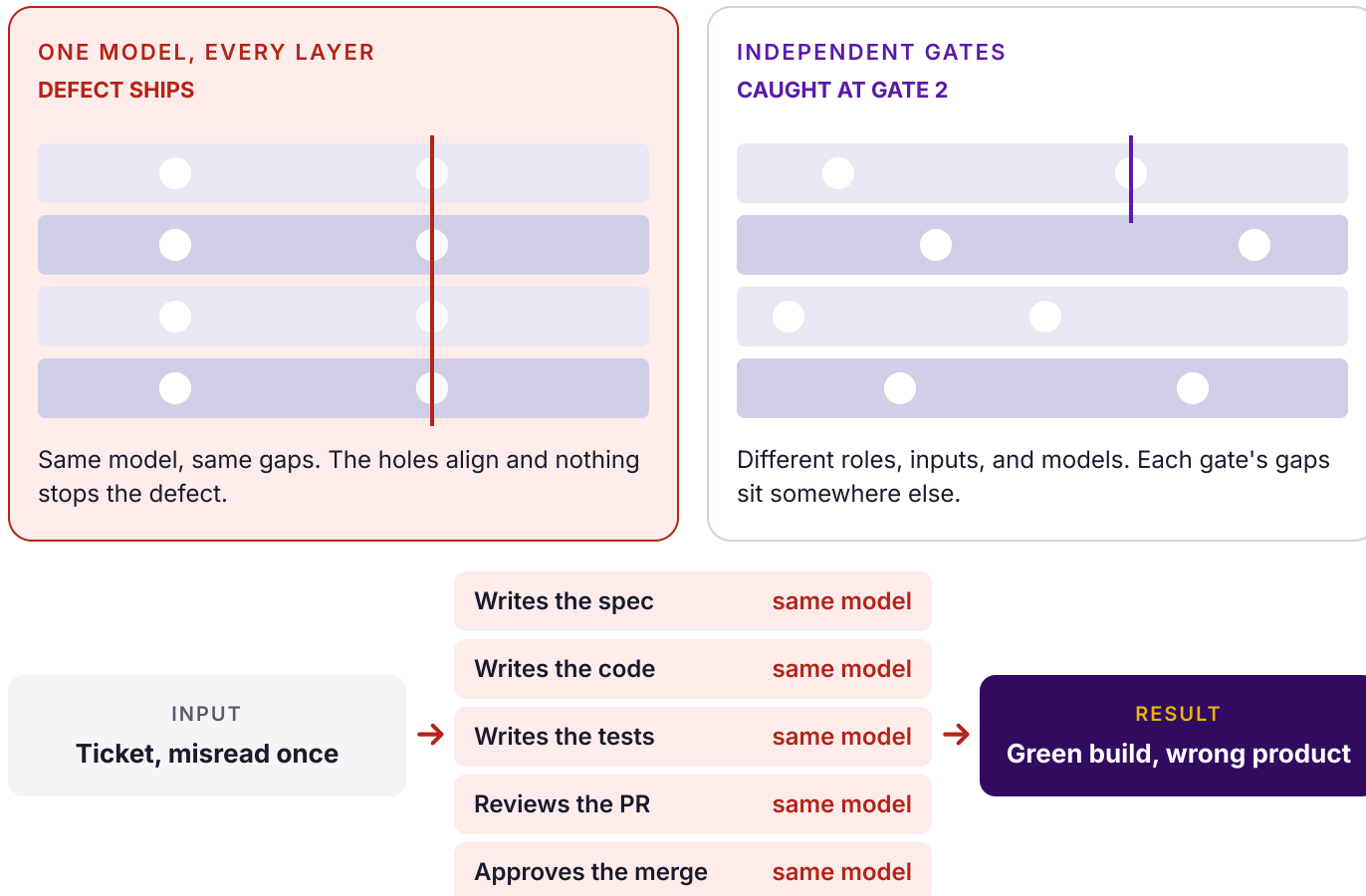
	SDLC · BUILT FOR PEOPLE	AGENTIC SDLC · BUILT FOR MODELS
Defends against	Forgetfulness, ego, fatigue	Stopping early, agreeing with the prompt, drifting, gaming checks
The spec	A requirements document people interpret	Acceptance checks, locked before any code is written
Review	Peer review: does a colleague approve?	Adversarial review: a separate reviewer tries to break it, and only reproducible findings count
Unit of trust	The engineer who wrote it	Evidence at every gate, plus two human sign-offs
Audit trail	Commit messages and ticket comments	A record of every gate result, readable by auditors
Cost over time	Process overhead compounds	Each finding becomes an automatic check for the next run

WHAT TRANSFERS

Version control, CI, and code ownership still matter. What changes is who checks, what they check for, and what counts as proof.

When one model runs every check, its blind spots line up.

Layers of checks work when each layer fails in a different place. A second engineer misses different things than the first. When the same model writes the spec, the code, the tests, and the review, every layer shares the same misunderstanding, and the defect passes straight through.



What a single model misses

It grades its own homework

Tests written by the author assert the author's reading, right or wrong.

It agrees with itself

Asked to review its own diff, a model tends to confirm the framing it started with.

It games its own checks

If the same agent can edit the test, the fastest way to green is to weaken the test.

A single model driving every check produces confidence, not evidence.

Eight agent mistakes your current process was not built to catch.

A useful test for any gate: which agent mistake does it catch? If nobody can answer, the gate is overhead. These eight patterns are why “the agent says it works” should never be a release criterion.

F1 Calling it done too early

Stops as soon as the code looks plausible and builds.

STOPPED BY Verifiable acceptance criteria

P1

F2 Agreeing with the prompt

Accepts the assumptions in the request, even wrong ones.

STOPPED BY Adversarial review P4

F3 Drifting in long sessions

Early decisions fade as the conversation gets longer.

STOPPED BY Fresh-context handoff P2

F4 Inventing with certainty

References APIs, files, or facts that do not exist.

STOPPED BY Risk review before build P1

F5 Gaming the check

Weakens a test or special-cases an input so the check passes.

STOPPED BY Mutation testing P3

F6 Stopping at a familiar number

Reports a habitual number of issues, however many there are.

STOPPED BY Seeded-defect checks P4

F7 Writing too much

Adds files and abstractions where a small change would do.

STOPPED BY Learnings become lint rules

P6

F8 Mixed-model seams

Passes work between models and leaves mismatched assumptions.

STOPPED BY One model per task P3

READ IT AS A MAP

Each mistake is caught at a specific step. The next page shows where those steps sit in the outer loop.

Seven steps, each with a gate at the exit.

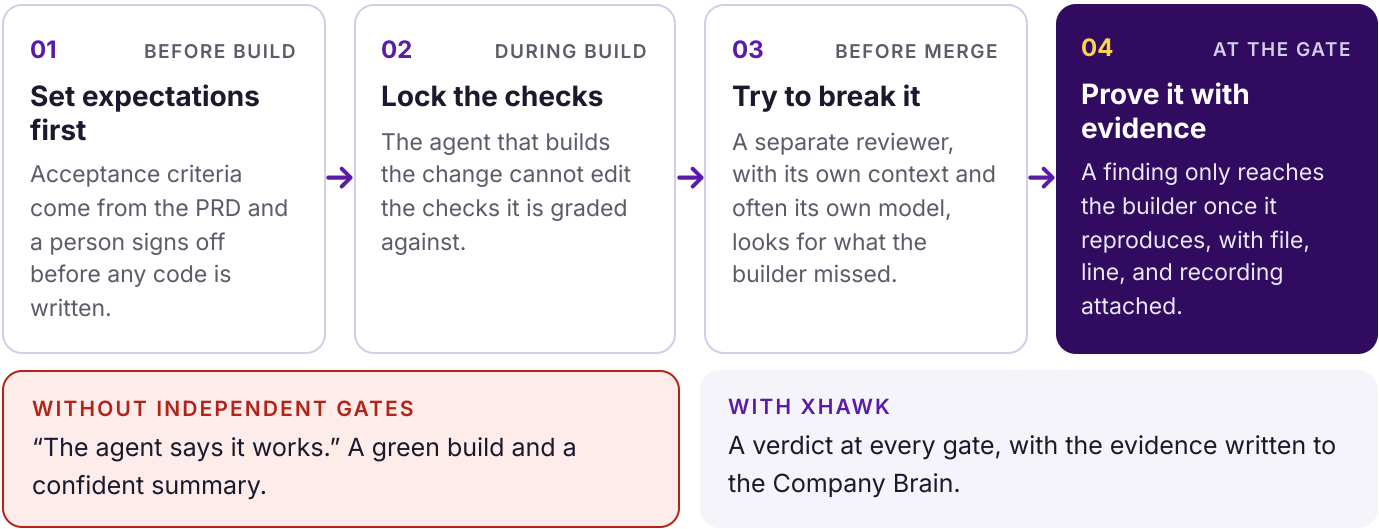
Every step in an agentic lifecycle should end with a clear exit check that leaves a result behind. Because each result is recorded, the whole lifecycle can run in CI instead of living in a process document nobody reads.

P0	Route	Classify the ticket by risk and send it down the right path.	✓ EVIDENCE
P1	Clarify	A person confirms the spec matches their intent, and each acceptance criterion says how it will be checked.	♦ HUMAN
P2	Plan	Each task is written clearly enough that a new agent with no history could pick it up.	✓ EVIDENCE
P3	Build	Acceptance checks are locked before coding starts, then the change is built to pass them, with one model owning each task.	✓ EVIDENCE
P4	Challenge	A separate reviewer looks for ways the change breaks. Only findings that reproduce are kept.	✓ EVIDENCE
P5	Accept	A person checks the change running in a real environment and accepts it.	♦ HUMAN
P6	Learn	Every finding is turned into a rule or check, so the next run catches it automatically.	✓ EVIDENCE
0	Passed	Result and evidence recorded	
1	Could not run	Fix the environment and run again	
2	Blocked	The gate ran and stopped the change	

TWO HUMAN GATES Senior engineers spend their time on two decisions: before the build, is this the right thing to build? After it, does the running change do what we intended? The other five gates run on their own.

If a gate never says no, it is not checking anything.

More verification does not mean more tests written by the same model. It means checks the builder has no way to influence: expectations locked before coding, tests proven to fail when behavior changes, and reviewers who are trying to find problems rather than approve.



How XHawk keeps verification independent of the agent that built the change.

Four checks a single model cannot give itself

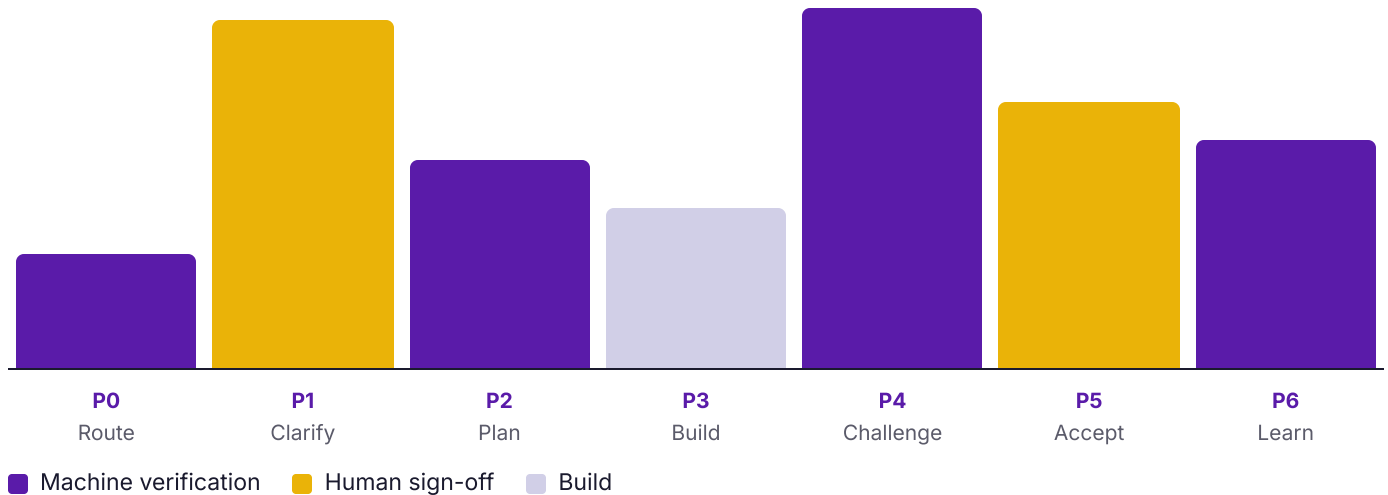
CHECK	WHAT IT PROVES	FAILURE IT STOPS
Locked checks	Checks existed before the code and were not changed since	Gaming the check
Mutation testing	The tests fail when the behavior changes	Hollow tests that always pass
Adversarial review	A separate reviewer tried to break it and found nothing that reproduces	Agreeing with the prompt
Seeded-defect checks	Review finds bugs that were added on purpose	Stopping at a familiar number

FIND PROBLEMS, NOT APPROVE

Traditional review asks a colleague for approval. Agent review should ask a separate reviewer to find what is wrong, and pass a finding to the builder only after it reproduces.

Put the effort into intent and verification, not the build.

In the traditional SDLC, writing code was the costly step, so most process wrapped around it. With agents, writing code is the cheap step. The expensive outcome is shipping the wrong thing, so effort moves to defining intent up front and verifying the result afterwards.



Illustrative effort by step. Build is the cheapest step; intent and verification carry the cost.

THE METRIC THAT MATTERS

Measure the cost of each verified change that reaches production, not AI spend per engineer.

Review keeps pace

Automatic gates run on every change, at a depth no human team could keep up.

Audit-ready by default

Every merge comes with a record of each gate result that compliance teams can read directly.

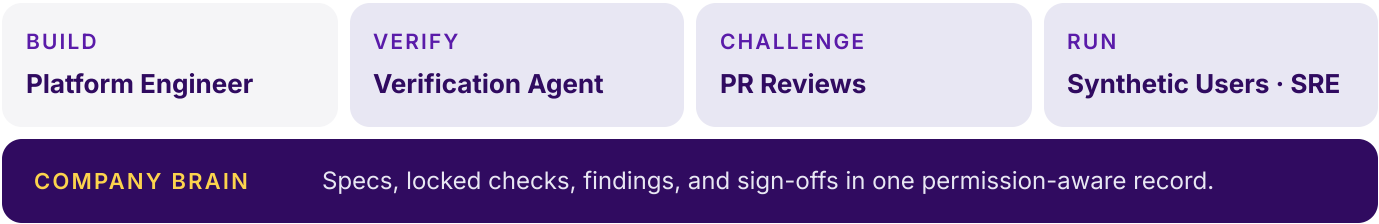
Gets cheaper over time

Each finding becomes a rule, so the same mistake is caught automatically on the next run.

Separate AI employees for separate gates.

XHawk runs the outer loop with AI employees that each own a role, so the agent that builds a change is never the one that approves it. Each gate writes its evidence to the Company Brain, which becomes the memory the next run starts from.

GATE	XHAWK AI EMPLOYEE	EVIDENCE IT LEAVES
P0 · Route	Product Manager	Risk tier and routing, linked to the ticket
P1 · Clarify	Product Manager, with a person	Spec with a verification method per criterion; human sign-off
P2 · Plan	Platform Engineer	Task plan a fresh agent can pick up with no prior context
P3 · Build	Platform Engineer, Verification Agent	Locked acceptance checks, and a PR that passes them
P4 · Challenge	PR Reviews, Synthetic Users Agent	Reproducible findings with file, line, and recording
P5 · Accept	A person, with SRE Agent	Sign-off on the running change; production checks
P6 · Learn	Company Brain	New lints, review rules, and journeys from every finding



Different roles, and where useful different models, at each gate. Humans own intent at P1 and P5.

START WITH ONE GATE

Most teams start at P4 with PR Reviews, then add locked acceptance checks at P3 once findings are trusted.

QUESTIONS FOR THE CTO

- 01 Is the model that writes your code also the one that reviews and approves it?
- 02 Can your coding agent edit the tests it is graded against?
- 03 When did a gate in your pipeline last block a change?
- 04 Which two decisions in your lifecycle still need a person, and is that where your senior engineers spend their time?

**One model checking
itself is confidence.
Independent gates are evidence.**

Gate your outer loop with XHawk.

AI Employees for Product, Engineering & Analytics.

Learn more →
xhawk.ai