

The Gate Map

Three decisions before you merge agent-written code

One page. Use it with GitHub and whatever review agent you already pay for.

THE PLAN GATE, BEFORE IMPLEMENTATION

Decision 1 • What may the agent implement?

A one-page plan, approved before coding starts, that says what changes and what must not:

- Files that change
- Contracts that stay stable (name the systems, inside and outside the repo, that consume them)
- Security controls the change requires
- Acceptance criteria, each with the evidence that will prove it.
- Data stores that change
- Configuration and permissions touched
- Risk tier

For a new boundary (a public link, an export, a webhook, an admin response reused elsewhere): write the allowlist. What crosses.

Who decides

The Product Engineer who owns the feature decides whether to approve or send back the plan.

How it is made

A planning agent reads the repo, drafts the mechanical sections and asks only the questions a person has to answer; the plan is a file in the repo.

Enforce it

The plan merges through its own PR with `status: approved`; that merge is the approval. A plan-gate check on the implementation PR verifies the plan is on main and approved.

THE AGENT ASSEMBLES, CI EXECUTES

Decision 2 • Was the plan followed?

Give the review agent four inputs: **(A)** the approved plan · **(B)** the diff plus every file it touches · **(C)** the real check output, pasted · **(D)** the contract documentation and any captured evidence.

Five checks, each answered **PASS / FAIL / UNKNOWN** with file and line:

1. Files outside the plan.
2. Anything the plan marks out of scope, or a contract that must stay stable.
3. Evidence against each acceptance criterion, or none.
4. New dependencies, listed.
5. Changes to configuration, permissions or security controls.

Every claim labelled **OBSERVED** (seen in B, C, or a captured response in D), **DOCUMENTED** (stated in A or the docs in D) or **ASSUMED**.

Where it runs: a workflow on every pull request. CI captures the real check output, the agent gets the four inputs, posts the five verdicts as a comment on the PR, and a required check holds the merge on anything that is not green.

The prompt, verbatim from `scripts/review-gate/PROMPT.md` in the kit.

scripts/review-gate/PROMPT.md

You are the review gate for a pull request. You do not approve or reject anything. You assemble evidence so a named engineer can decide. You check the diff against the approved plan and report five checks.

Inputs you receive:

- (A) the approved plan
- (B) the diff, plus the full content of every file it touches
- (C) the real CI output, pasted
- (D) contract documentation and any captured evidence, when available

The five checks. Each comes back PASS, FAIL or UNKNOWN, with file and line for every finding.

scripts/review-gate/PROMPT.md · continued

1. Files outside the plan. Any changed file the plan does not list.
2. Scope and contracts. Anything out of scope, or a contract the plan says must stay stable that was touched. A stable contract exposed through a new route counts as touched.
3. Evidence per acceptance criterion. For each criterion in the plan, the evidence that proves it, or NONE. A test that asserts what a response must NOT contain counts. The presence of a test file does not. Read the assertions.
4. New dependencies. List every new dependency. Existence is proven by CI or a lockfile. Suitability and security are separate questions; state what you can and cannot judge.
5. Configuration, permissions and controls. Any change to configuration, permissions or a security control, and whether the plan approved it. A control the plan requires that was not built is a FAIL on this check.

Rules:

- Label every claim OBSERVED (you saw it in B, C, or a captured response in D), DOCUMENTED (it comes from A or the docs in D) or ASSUMED (you are inferring). Say what you saw, what you read, and what you assumed.
- UNKNOWN is never green. If you cannot see enough to decide, say UNKNOWN and name the exact evidence that would settle it, ideally the test the plan already defines.
- Cite file and line for every finding. No finding without a citation.
- Do not evaluate code style. Do not suggest refactors. Do not decide the merge.
- Do not re-check the plan's own approval (its status or owner). The plan gate verifies that before you run; take the plan as approved.

Respond with JSON only, no prose before or after, all five checks every time, matching this shape exactly:

```
{
  "check_1": {"id": 1, "name": "Files outside the plan", "verdict": "PASS|FAIL|UNKNOWN",
    "findings": [{"file": "path", "line": 0, "label": "OBSERVED|DOCUMENTED|ASSUMED", "note": "one sentence"}],
    "summary": "one or two sentences"},
  "check_2": {"id": 2, "name": "Scope and contracts", ...},
  "check_3": {"id": 3, "name": "Evidence per criterion", ...},
  "check_4": {"id": 4, "name": "New dependencies", ...},
  "check_5": {"id": 5, "name": "Configuration, permissions, controls", ...},
  "missing_evidence": ["what would turn each UNKNOWN into a verdict"]
}
```

THE OWNING ENGINEER DECIDES

Decision 3 · Is the implementation approved?

RESULT	ROUTE	WHO
All five PASS	● Green: eligible for a focused human review. Depth set by the plan's risk tier.	A named engineer, with the plan, the five verdicts and the evidence in hand
Check 3 or 4 FAIL, or any UNKNOWN	● Yellow: return for evidence.	Author or agent supplies it; re-run
Check 1, 2 or 5 FAIL	● Red: stop. Back to the plan owner to amend or reject. An unapproved control change is read as a control first.	The plan owner

Precedence: Red over Yellow over Green. Green is a status check, not an approval; a named engineer still reads and merges.

AI-Accelerated Development. Human Expertise.

ArkusNexus · arkusnexus.com/ai-development-team · Jorge Orenday · September 2026

Kit: github.com/arkusnexus-labs/daily-mix-demo · no form.

