AI Deliberation & Claude Code

/second-opinion

Called from the terminal you are already in. Your agent keeps the repository; a panel of engines from other labs attacks the answer — and you see where they split.

A WORKFLOW GUIDE

QUORUM.DOG

What is in here

Sections 03 to 05 are the argument. 06 and 07 are what you commit and what you type. 08 to 11 are the work itself, with the exhibits. 12 and 13 are wiring. If you are short of time, read 06 and 08.

03

What Quorum is, in one page.

Quorum is a deliberation platform. One question goes to several frontier models from different labs at once. They answer independently, critique each other, and are judged — and what comes back includes **where they disagreed**.

Claude Code is very good, and it is one model. Asking it to check its own work gets you the same priors twice, in a more careful tone. Models trained elsewhere fail in different places. That is the whole mechanism, and there is nothing else to it.

A SEAT

One frontier model on the panel. The standard panel seats three, reasoning independently. Quorum represents them as shapes rather than naming vendors.

A MODE

The configuration of a deliberation — which engines sit, how many rounds, how deep. Not a knob on a single model: a repeatable decision setup you select.

A RECEIPT

The record of a completed deliberation: which model sat in each seat, judge scores, difficulty, cost, latency. §11.

WHERE IT RUNS

Quorum's own surfaces, an OpenAI-compatible REST API, and a hosted MCP server — which is how it reaches your terminal.

THE CLAIM, STATED NARROWLY

Quorum does not claim your agent is bad. It claims that a single model, however capable, **cannot be its own second opinion** — it has one way of being wrong, and it applies that way consistently. It is why "review your own patch" and "are you sure?" come back with the same answer in a better mood.

AND WHERE IT DOES NOT APPLY

Most of what you ask an agent is execution, not judgement. A panel is deliberately wrong for renames, formatting, syntax and writing a test for code that already exists — and Quorum's own `estimate` call will say so before you spend anything (§05, §13).

WHAT YOU WAIT FOR · MEASURED ON PRODUCTION, 2026-08-20

CALL	TIME	
`estimate`	~0.9 s	Classification only. It never convenes anything, and it is free.
`deliberate · median (p50)`	28.5 s	n=3,845 deliberations.
`deliberate · p90`	88.2 s	Which is why a panel belongs on a branch you chose, not on every turn — and never inside a loop you are iterating every few seconds.

04

/second-opinion

Your agent is good. Give it somewhere to go when it isn't sure.

Claude Code reads the repository, runs the suite, writes the patch and tells you what it did. On most of what you ask it, that is the whole job.

Then there are the turns where it is **producing** an answer rather than knowing one — why the assertion still fails after three fixes, whether this is the schema you want to live with. It answers those too, fluently and in one voice, because one voice is what one model has.

One point worth being precise about: **your agent is not on the panel.** It stays where it is — holding the repository, the test output and your standing instruction — and calls in independent models, engines from other labs among them.

The division of labour follows from that. **Claude Code has your codebase; Quorum has your paragraph.** So you do not escalate the repository. You escalate the reasoning: the diagnosis it reached, the interface it proposes, the hunk you are staring at.

```
> why does test_reconcile_pending still fail after the cursor fix?

  Read(src/ledger/reconcile.py)              214 lines
  Bash(pytest tests/test_ledger.py -q)       1 failed in 0.42s

  Three diagnoses proposed, assertion unchanged. Judgement call,
  not a lookup. Escalating rather than trying a fourth.

  quorum - estimate                          0.9 s
    deliberation_value  likely_helps    basis  hypothesis

  quorum - deliberate                       31.4 s
    one answer, plus where the seats split

  Two seats attacked the code. One attacked the test - unreconciled.
```

Exhibit 01 — One turn in the terminal, with the panel in it. Only the judgement branch convenes it; everything else the agent answers directly, with no panel and no wait. Session transcript is illustrative.

DIFFERENT PRIORS

Asking twice gives you **one opinion, twice**.

BEFORE THE MIGRATION

It arrives while the schema is **still a proposal**.

CODE YOU DID NOT WRITE

A read from **not the thing that wrote it**.

AGREEMENT COUNTS

Landing together is **information too**.

05

The five moments it escalates on.

Not one language and not one stack. These recur in any codebase old enough that being wrong is expensive — and in every one of them the mechanical work is finished and the judgement is what is left.

01	**A loop that will not close**	Three fixes, three confident explanations, and the same assertion failing. The fourth attempt from the same model will draw on the same priors as the first three. Hand the panel the diagnoses, not the bug. **§08 works this one end to end.**
02	**A decision with a migration behind it**	A schema, an auth model, the shape of a public endpoint. No code to run and no test to fail, which is exactly where a confidently wrong single answer is expensive and where independent models genuinely disagree. **§09.**
03	**A large diff you did not write**	The suite is green. Green tells you it runs. It does not tell you this is the change you wanted, and it cannot tell you what the change quietly made true. **§10.**
04	**A plan you are about to let run**	Before you turn an agent loose across forty files, the plan is a paragraph and arguing with it costs a minute. Afterwards it is a branch, and disagreeing with it costs an afternoon of unpicking.
05	**Code you inherited**	A service whose author left, a vendor SDK, a subsystem nobody owns. Your agent can read all of it; what it cannot do is disagree with itself about which part is load-bearing.

THE COMMON SHAPE

Each one is a moment where the code is not the hard part. That is the line your standing instruction draws in the next section, and it is the only line it needs to draw.

AND WHAT IT SHOULD NEVER ESCALATE

Renames, formatting, syntax, a test for code that already exists, or anything you are iterating on every few seconds. Quorum's classifier agrees: `estimate` returns `likely_hurts` when the answer is a verbatim artifact a synthesis pass would only rewrite — which describes most patches. **Ask the panel to attack the approach; write the code yourself.**

06

Teach it once.

Give your agent a standing rule for when to convene the panel, and the decision stops being yours to remember. This is the whole setup.

CLAUDE.md is where Claude Code keeps standing guidance for a project, and that is where this belongs. Commit it and the rule is the repository's rather than yours — it survives the session, it applies to everyone who clones, and it can be argued with in review like anything else you commit.

CLAUDE.MD — PASTE AS WRITTEN, THEN EDIT THE LAST LINE

```
You have a Quorum MCP server. It convenes a panel of frontier models
from several labs and reports where they disagree.

Use it on judgement, not on work you can do yourself.

Convene the panel when:
  - you have proposed a diagnosis twice and the failure has not
    changed;
  - I am choosing a schema, an auth model, a public interface, or
    anything with a migration behind it;
  - you are about to change many files I would have to unpick;
  - I ask you to review a change that neither of us wrote;
  - I ask what we have not thought of.

When you do:
  - send the reasoning, not the repository: the diagnosis, the
    constraint it must satisfy, the option I am leaning toward;
  - paste the actual error, signature and version pins, not your
    description of them;
  - report where the seats disagreed and do not reconcile it for me;
  - never ask the panel to write the patch. Ask it to attack the
    approach, and I will write the patch.

Do not convene it for renames, formatting, syntax, tests for code
that exists, or anything I am iterating on. Answer those yourself.

Escalate without asking me first.   ← or "ask me first"
```

BEST PRACTICE

Commit it, and let it escalate on its own. The point of this setup is that it catches the turn you did not notice was a judgement call. If you have to remember to invoke it, you will remember exactly when you least need it. Approving each escalation is right for week one and wrong after that.

THE ONE LINE THAT MATTERS MOST

"Send the reasoning, not the repository." The panel argues about what it is given, and it is given a paragraph. An agent left to summarise will write *a race condition in the ledger writer*; you want the assertion, the two call sites and the fact that three fixes did not move it.

07

Steering it in the moment.

The standing rule handles the turns you did not spot. These are for the turns you did — when you already know the question is worth an argument and you want one now.

"You have fixed this twice and it still fails. Send the panel your three diagnoses and ask what all of them assume."

Attacks the reasoning, not the bug

"Before you run this across forty files — put the plan to the panel and give me the strongest argument against it."

A minute now against an afternoon later

"This migration is one way. Convene the panel and tell me what I am trading away."

Reframes a decision as a trade

"Review this diff with the panel. I did not write it, and the tests passing is not the question."

Green is not a review

"Where did the seats disagree? Do not reconcile it — show me the split."

The follow-up worth making a habit

"Send them the actual trace and the version pins, not your summary of them."

When precision is the whole question

BEST PRACTICE

Use both. The standing instruction is the safety net; these are the times you reach for it deliberately. If you only ever do one, do the standing instruction — the escalations you would have typed by hand are the ones you already knew were hard.

WHAT MAKES AN ESCALATION LAND

Name the error. Say what breaks if the answer is wrong. Say what you are leaning toward — a panel given a position to attack argues far harder than a panel asked an open question. And ask for the split explicitly; a tidy synthesis is the least useful thing a panel can hand you.

08

LOOP → DESIGN → DIFF

The loop that will not close.

The same assertion has failed for forty minutes. Your agent has proposed three fixes. Each was confident, each was plausible, and the number in the failure has not moved.

IN THE TERMINAL

```
$ pytest tests/test_ledger.py -q -k reconcile
F                                        [100%]

    assert len(ledger.pending) == 4
E   AssertionError: assert 3 == 4

1 failed, 41 passed in 0.42s
```

ILLUSTRATIVE OUTPUT

WHAT YOUR AGENT HAS ALREADY SAID

1 Off-by-one in the pagination cursor. *Patched. Still 3.*

2 The fixture seeds three rows, not four. *Checked. It seeds four.*

3 `reconcile()` filters on status; loosen the predicate. *Patched. 5, then 3 again.*

Three plausible answers, one set of priors. A fourth attempt is the same reasoning at a different angle.

What you send is not the repository. It is those three diagnoses, the assertion, and the one fact the agent cannot weigh: that all three failed. The panel is being asked to attack a piece of reasoning, which is a paragraph-sized object — not to read a codebase, which is not.

WHAT THE PANEL RETURNS

- ▲ **All three assume the count is read once.** If the value is read after a flush that nothing awaited, no change inside `reconcile()` can move it — which is consistent with three fixes producing three different numbers.
- ■ **Not timing. The test.** All three fixes take the expected 4 as given. Nobody has re-derived that number since the deduplication rule changed, and 3 may be the correct answer.
- ● **Neither can be settled from what was sent.** The missing fact is whether the failure is deterministic. Three fixes moving the number 3, 5, 3 is the tell, and nobody has run it twenty times.

The seats did not vote. Two attacked the code and one attacked the test — and the third objection costs nothing to check and would have ended the afternoon at minute five.

ELAPSED

A panel takes a median of **28.5 seconds**, 88.2 at the ninetieth percentile. You have already spent forty minutes.

P50 28.5 S · P90 88.2 S · N=3,845 · 2026-08-20

Exhibit 02 — A failing test, three rejected diagnoses, and what three seats said about them. Output and seat positions are illustrative of the pattern, not a transcript of a specific session.

09

LOOP → **DESIGN** → DIFF

The decision you cannot cheaply reverse.

A schema. An auth model. The shape of a public endpoint. There is no code to run and no test to fail here, which is precisely why one confident answer is the dangerous kind.

THE DECISION, WRITTEN DOWN

```
Multi-tenant, one deployment. Two shapes:

A   tenant_id on every table, one schema
    + cheap to run, easy to query across tenants
    - one bad WHERE clause crosses customers

B   schema per tenant
    + isolation is structural, not a predicate
    - 400 migrations per change, and churn

Contract: per-tenant deletion inside 30 days.
40k rows in events today. Either way, the
migration runs one way.
```

ILLUSTRATIVE

THE MOVE

Do not ask which is better. Name the one you are leaning toward and ask each seat to argue against it and say what it is trading away.

ESTIMATE FIRST, IT IS FREE

`estimate` takes about 0.9 s and returns `deliberation_value` before it returns a price: `likely_helps`, `likely_hurts` or `unknown`. It carries `basis: hypothesis` and an evidence line — a per-task pattern from n=36, not a measured effect — so read it as a hint. It is still a better gate than price.

WHERE THE PANEL SPLIT

▲ **A, and make the isolation structural inside it.** The leak risk is real and B does not fix it so much as move it; row-level policy plus a test that runs unauthenticated does. B trades a query problem for an operations problem you have on every deploy.

■ **Same answer, different reason.** A wins on the deletion clause, not on cost: deletion inside 30 days is one predicate job against one schema, and 400 schemas make it 400 jobs that can each fail separately at 3am.

● **Neither, on this evidence.** 40k rows does not force the choice. What is actually being decided is which mistake is cheaper to unwind in a year, and nothing in the brief says how many tenants there will be.

Two seats agreed on the answer and disagreed on why. The reason is the part you write down, and the part the next person will argue with — a bare verdict would not have survived the first “why A?”

WHAT YOU DID NOT SEND

Not the schema, not the migration, not the repository. Twelve lines describing the choice and the constraint. Put any fact the answer turns on *into* those twelve lines.

Exhibit 03 — A one-way decision, and where the seats split. Options and seat positions are illustrative of the pattern, not a transcript of a specific call.

10

LOOP → DESIGN → **DIFF**

The diff you did not write.

Nine hundred lines, produced in one run, suite green. Green tells you it runs. It does not tell you this is the change you wanted, and it will never tell you what the change quietly made true.

ONE HUNK OF IT

```
--- a/billing/invoice.py
+++ b/billing/invoice.py
@@ -118,9 +118,9 @@ class InvoiceRun:
-        with self.db.transaction():
-            for line in lines:
-                self._apply(line)
+        for line in lines:
+            self._buffer.append(self._row(line))
+        self._flush()
@@ -204,6 +204,9 @@ class InvoiceRun:
+    def _flush(self):
+        self.db.execute_many(self._buffer)
+        self._buffer.clear()
```

ILLUSTRATIVE DIFF

WHAT THE COMMIT SAYS

"Speed up invoice runs." It is true, and it is one of two changes in the hunk.

WHAT NO TEST ASSERTS

That a failed line leaves the run with nothing written. Nothing was testing atomicity before, so nothing broke when it left.

WHAT THE PANEL RETURNS

▲ **Two changes arrived as one.** The batching is defensible; dropping the transaction boundary is a separate decision that nobody made explicitly. Split the commit and the second half becomes reviewable.

■ **Disagrees that atomicity is gone.** On some drivers `execute_many` is a single statement and the guarantee survives. The defect is that the diff does not say which driver, so the reviewer cannot know either.

● **Both are arguing about correctness.** The durable cost is the commit message: filed under performance, so a bisect for a consistency bug in six months will not stop here.

Three seats, three different problems with one hunk, and only one of them is a bug. The other two are things a green suite structurally cannot tell you — and neither can the model that wrote it.

GIVE IT THE INVARIANT

The panel sees the hunk you paste. Name the invariant you think is at risk, and it argues about that. Paste the hunk alone and you get three code reviews.

Exhibit 04 — A diff, and three different problems with it. The diff and the seat positions are illustrative of the pattern, not a transcript of a specific call.

11

The receipt, and what you can do with it.

Every deliberation leaves a record. It is the difference between “the AI said” and a decision you can hand to whoever is on call in six months.

WHAT GET_RECEIPT RETURNS

request_id	rq_8f2c41a9
mode	quorum-standard
difficulty	0.78
seat 1	judge score, model
seat 2	judge score, model
seat 3	judge score, model
cost	billed, per call
latency	31.2 s

SHAPE OF THE RESPONSE · ILLUSTRATIVE VALUES

The judge scores are the part people underuse. They tell you whether the seats **actually disagreed** or converged on the first pass — which is how you find out that a question never needed a panel.

Quorum names seats by shape rather than by vendor on anything shareable, so a receipt can be pasted into a public pull request without carrying somebody else’s trademark.

IT EXPORTS

From Quorum, a completed deliberation renders as a full transcript PDF, a carousel, a card or a short video — the seats, their positions and the split, laid out.

IT TRAVELS

That artefact is what goes in the pull request, the decision record or the incident write-up. The argument travels with the commit instead of evaporating in a terminal you have since closed.

IT TUNES

Receipts that keep showing immediate convergence mean your threshold is too low — you are paying for agreement. §13.

WHERE THIS IS GOING, STATED AS DIRECTION AND NOT AS A FEATURE

A receipt records one deliberation. What it does not yet do is accumulate: the decision, the split, what was chosen, and eventually whether the thing you were warned about actually happened. A repository that kept that would have a decision record that knows how its own past calls turned out. Quorum does not do this today — there is no store, no outcome linkage and no retrieval — and it is named here as the direction the receipt makes possible, not as something you can switch on.

Exhibit 05 — A receipt.

12

Connect it.

A hosted MCP server. Nothing to install, nothing running on your machine: a URL and a bearer token is the whole integration.

Get a key from your organisation dashboard. Then ask your agent to call `list_modes`: free, requires a valid key, returns the panels you are entitled to — so one real answer proves URL, key and wiring together.

MCP CONFIGURATION · AND A CHECK YOU CAN RUN BEFORE INVOLVING THE AGENT

```
{
  "mcpServers": {
    "quorum": {
      "type": "http",
      "url": "https://www.quorum.dog/mcp",
      "headers": { "Authorization": "Bearer qk_..." }
    }
  }
}

curl -sS -X POST https://www.quorum.dog/mcp \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer qk_..." \
  -d '{"jsonrpc":"2.0","id":1,"method":"tools/list"}'
```

THE SIX TOOLS THAT APPEAR

TOOL	COST	WHAT IT DOES
`estimate`	Free	Classifies a question and prices it without running it.
`list_modes`	Free	Which panels your key may convene. Also the wiring check above.
`deliberate`	Paid	Convenes the panel. The one that costs and the one you wait for.
`get_receipt`	Free	For a finished call: seats, judge scores, cost and latency.
`run_test`	—	Mode testing. Added to the hosted server 2026-08-22.
`certify_mode`	—	Mode certification. Both drive Quorum's own testing, not your work.

TWO THINGS THAT LOOK LIKE FAULTS AND ARE NOT

A **GET** to that URL returns 405 — the server is Streamable HTTP, POST only, and that is the correct response. And the connection is stateless, so there is no session to keep warm.

THE WRAPPER KEY DIFFERS BY EDITOR

Claude Code and Cursor nest servers under `mcpServers`; VS Code uses `servers`. Same protocol, same URL, and the wrong key fails silently. The same token works against the REST API at `quorum.dog/v1`.

13

When the review runs without you.

Everything so far assumes you are at the keyboard, and for one developer working through one hard decision that is the right assumption — when the question genuinely matters, the price of the call is not what you are weighing.

Automation changes the arithmetic. An agent that reviews every pull request, or works unattended overnight, is making the escalation decision hundreds of times with nobody present to judge which of them deserved a panel.

That is what **estimate** is for. It classifies and prices a question without running it, free on every call and about 0.9 s, so an unattended process can decide for itself what is worth escalating.

The threshold is yours and belongs in your own code. Quorum's job is to hand that branch the facts it needs, at no cost.

WHAT ESTIMATE HANDS BACK

FIELD	WHAT YOU BRANCH ON
`deliberation_value`	`likely_helps` / `likely_hurts` / `unknown` — branch on this first. Carries `basis: "hypothesis"` and an evidence line: a per-task pattern from n=36, not a measured effect.
`difficulty_score`	0 to 1. Set your threshold from your own measured distribution, not ours.
`task_type`	What kind of question it is — how you exempt whole categories at once.
`estimated_price_usd`	What the real call would cost, so a budget ceiling can sit in front of it.
`billed_usd`	Returns 0 on every estimate call, by design — not as a trial allowance.

FAIL OPEN, ALWAYS

If the classification call fails, answer with your single model rather than erroring or defaulting to the expensive path. A router that breaks the build when one endpoint has a bad minute is worse than no router.

TUNE IT WITH RECEIPTS

If receipts on escalated items keep showing the seats converging with no dissent, raise the threshold — you are paying for agreement. If reviewers keep pushing back on single-model answers, lower it.

Start with CLAUDE.md and one repository you already care about. You will know inside a week which of the five moments is yours.

WHERE TO GO NEXT

The escalation router guide at `quorum.dog/docs` covers the code version of this branch in full, including how to pick a starting threshold.