I build AI you can trust *enough to act on.*

A working agentic AI system for equity analysis:
gated where it matters, graded on every run.

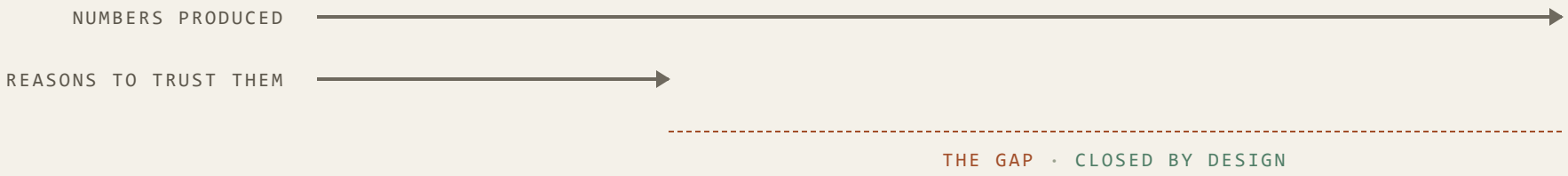
Eric Cohen product manager · a career where numbers face audits cohenemc@gmail.com

AI produces numbers far faster than it produces reasons to trust them. *Closing that gap is the product.*

It does one job: read a stock against the companies it truly competes with, and judge whether the price is out of line.

The user is an investor, and the answers end in buy, sell, or pass. At the moment of decision, a number is either proven or merely plausible.

The bet behind this system is that proof can be designed, not hoped for. Every slide that follows is a piece of that design.



I put a human in the loop *on every call the AI makes.*

I built the agent to do the analysis, so the investor is free to focus on the decisions.

Gating every step would mean doing the analysis twice. Gating none would mean trusting the AI with the answer. I drew the line at consequence: the AI reads, computes, and builds unattended, and the calls that shape the answer stop until a human locks them in. The result is a powerful engine that never finishes unchecked.

0 RUNS FINISH WITHOUT A HUMAN SIGN-OFF

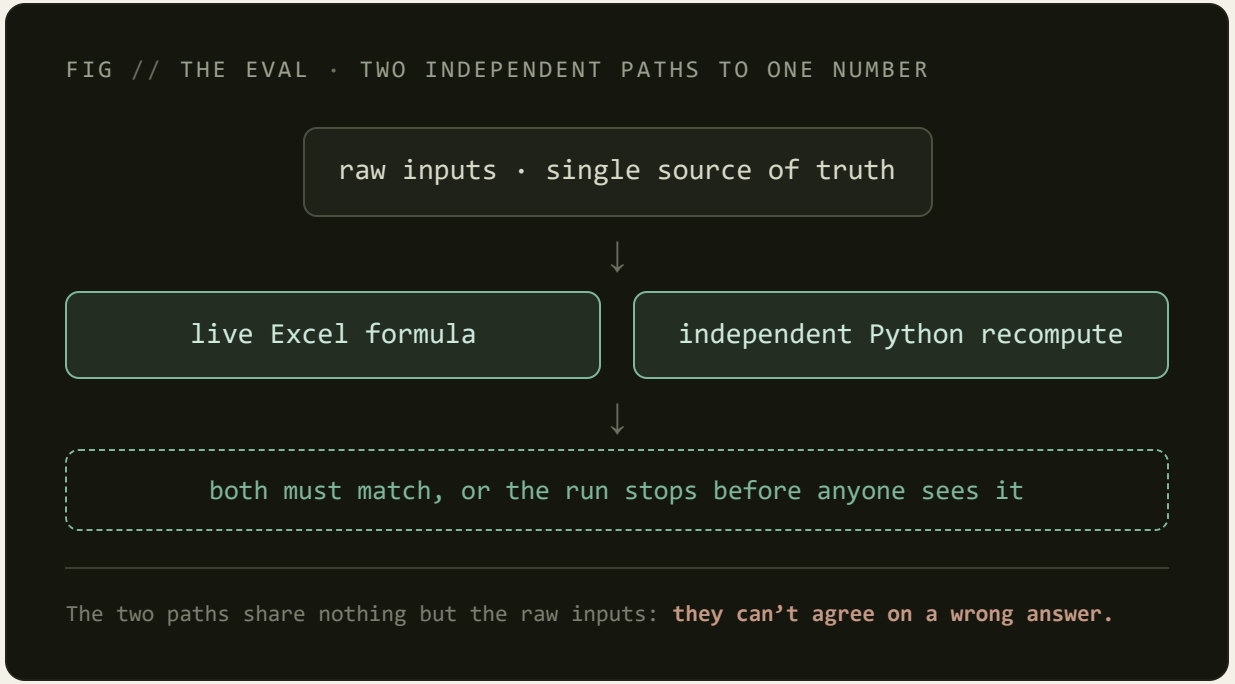
THE PRODUCT CALL *A capable AI, kept under a human’s command. Designing that balance is the* **AI-PM job.**

FULL CASE STUDY ↗

I make every number *safe to act on.*

An investor acts on these numbers, and a wrong one costs real money. So I don't trust AI output, only proof it can't fake.

The AI wrote the first version of this proof, and it graded its own work against a copy of itself. Every test came back green; one number still looked wrong. Both couldn't be true. I caught the circle, re-architected the proof, then planted bad math to confirm the rebuilt check fails exactly when it should.



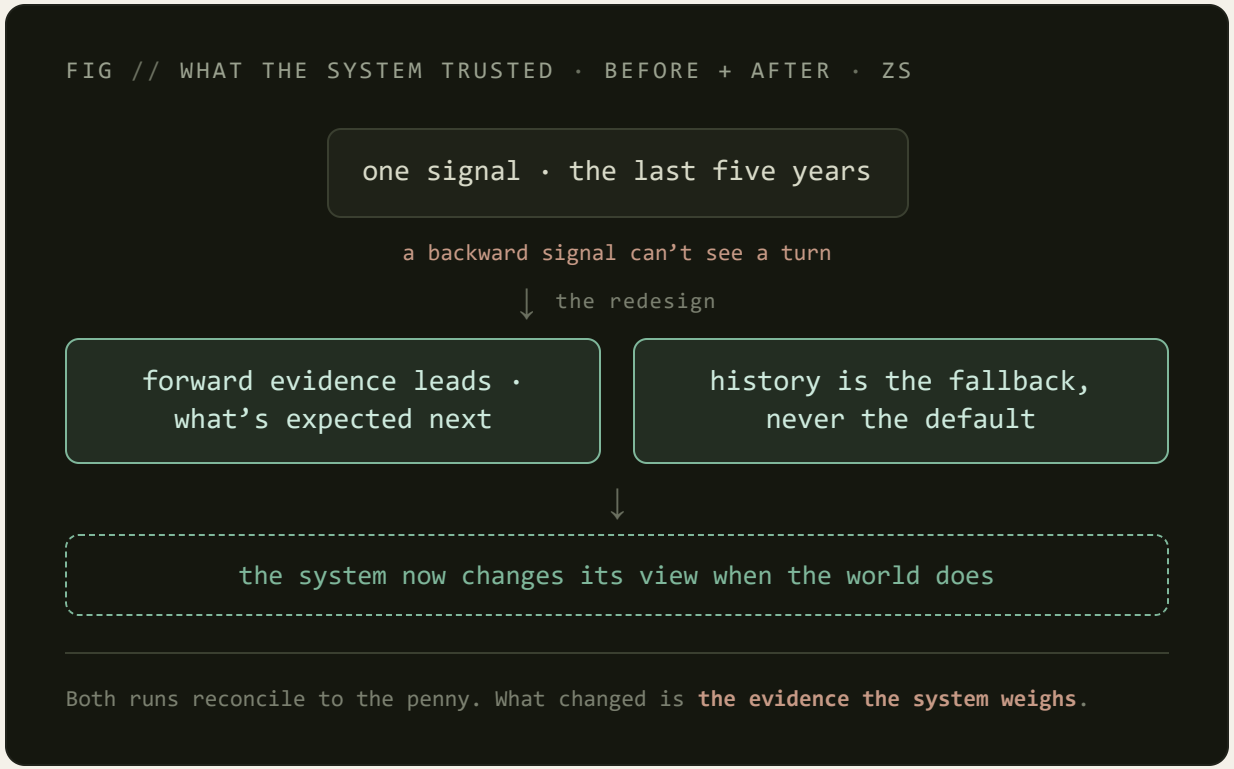
0 WRONG NUMBERS REACHED ME ACROSS 23 RUNS

THE PRODUCT CALL *An AI feature isn't done when it works. It's done when the user can* **trust** *it, so I build the proof in.*

I fixed a whole class of wrong answers *without touching the math.*

Every check in this system can pass and the call can still be wrong. Catching that takes judgment. Fixing it takes design.

One run read a stock as 341% undervalued. No heavily covered company trades at a quarter of its worth, so I pulled the call and traced the system's confidence to its source: five years of 40%-a-year growth had ended, and it was still pricing the old pace. I rebuilt what growth means to the engine.



+341% → +27% IMPLIED UPSIDE ON THE SAME STOCK, BEFORE AND AFTER THE REDESIGN

THE PRODUCT CALL

Evals are necessary, not sufficient. When judgment catches what they can't, I **build the judgment in.**

The AI never gets a blank page.
I designed what it's allowed to recommend.

Every run ends in a written recommendation: the conviction call the investor acts on. Those words carry more weight than any number in the model, so they face a test of their own.

This system once wrote a summary figure that existed nowhere in the data. That is the defining risk of generative AI, and it surfaced in my own product. I caught it and split the writing job: code now produces every number in the summary, and the AI's recommendation must clear a data check before the run can finish.

100% OF CONVICTION CALLS ARE TESTED AGAINST THE DATA

THE PRODUCT CALL

The riskiest AI output is the one a user believes. So the last thing the system checks is **the first thing the user reads.**

FULL CASE STUDY ↗

I made the agent more reliable as it did more, *by loading fewer rules at once.*

The more rules an agent holds at once, the worse it follows them. Reliability isn't about adding rules. It's about which one the agent is holding the moment it acts.

A skipped rule leaves no mark. A wrong answer comes back looking as finished as a right one.

A single rulebook loaded every run was the right call when the system was small. As it grew, the bottleneck became attention. So I split the rulebook and put a gate on the load. An instruction to load first is a request the agent can miss. A gate is not.

56% LESS CONTEXT CARRIED INTO EVERY RUN

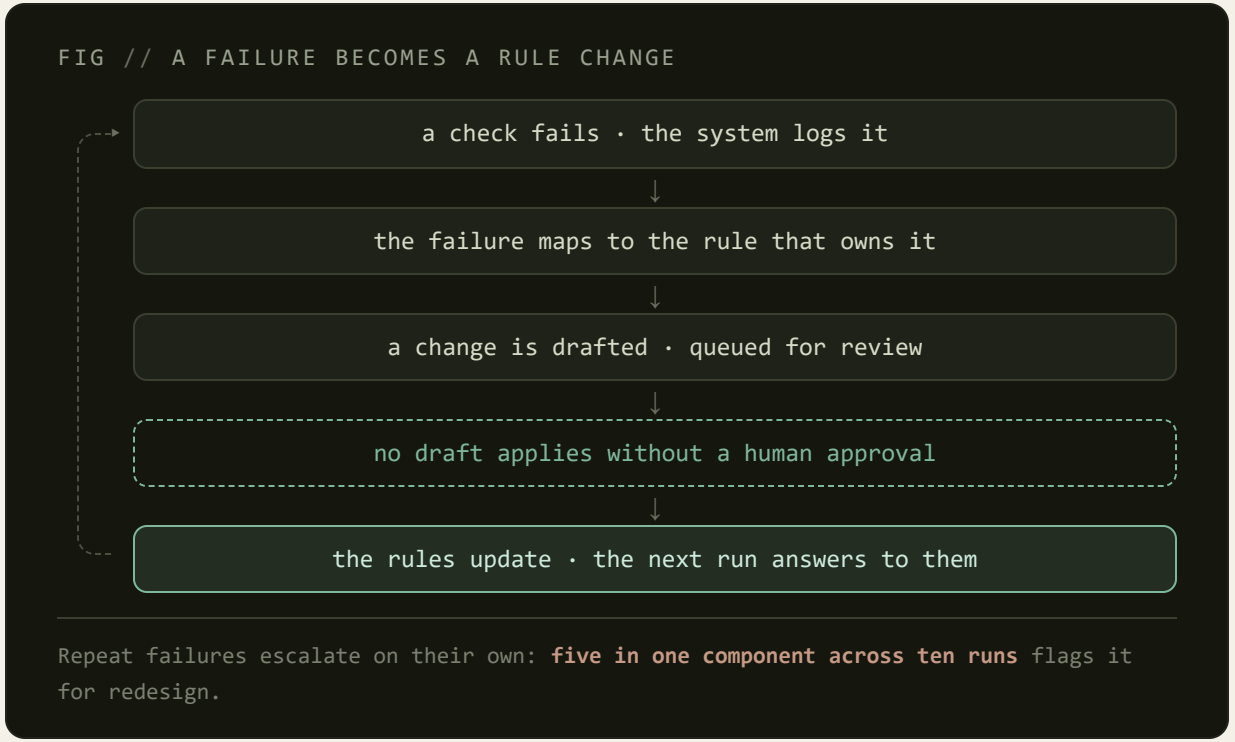
THE PRODUCT CALL

A reliable agent is **engineered**, *not prompted. I design what it holds so the rule that matters can't get lost.*

This system logs its own failures and drafts its own fixes. *Nothing changes without me.*

A product improves at the speed its failures become fixes. So I built the loop that turns one into the other, and pointed it at the system's own rules.

The tempting build was an LLM that reads each failure and rewrites its own rules. I built the boring version on purpose: deterministic drafting from a map I designed, predictable and auditable. The AI doesn't apply its own edits today. Whether it ever should is a risk decision, and evals are what make it a decision instead of a guess.



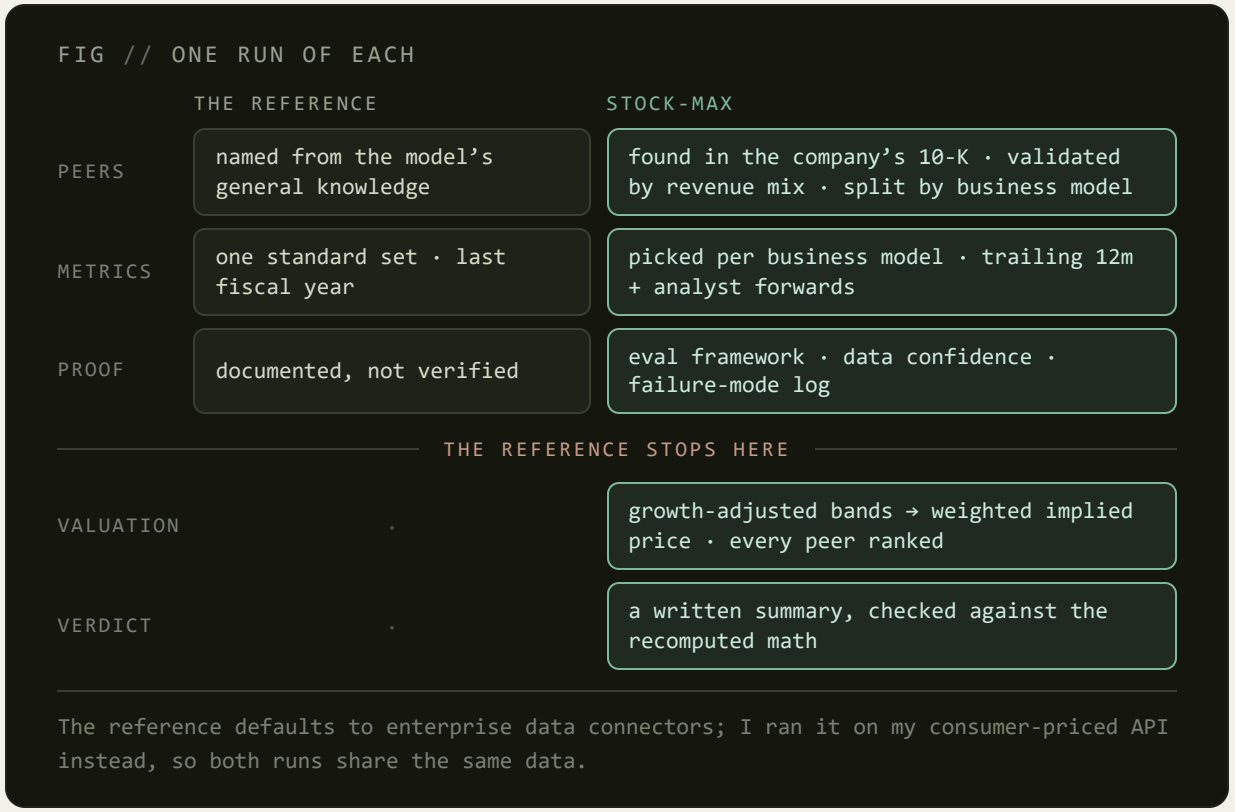
3 AI SKILLS COMPOSED INTO THE BUILD:
TEST DESIGN · DEBUGGING · CODE REVIEW

THE PRODUCT CALL *This is how one person ships at team speed:* **delegate the work, never the judgment.**

Institutional-grade output, *on consumer-priced data.*

Anthropic ships an open-source comps reference, deliberately bare, built to be tailored. I ran it side by side with my system and kept its output as my baseline.

An enterprise data contract never made sense for one person. My bet was that it didn't have to: I could engineer trust into the methodology layer instead of buying it from a data vendor. The delta in the figure is that bet, measured.



$0 SPENT ON ENTERPRISE DATA CONNECTORS

THE PRODUCT CALL — *The data sets the floor. The methodology sets the ceiling, and the ceiling is the part you* **design**.

An AI product you can trust is run by its numbers. *These are mine.*

The system is designed to catch its own flaws and improve run over run. This page is the record it keeps on itself.

One side of this ledger proves the system does its job. The other is the work: what shipped, what broke, and the eight items still open, one major, prioritized and worked like any product backlog. And every failure on this page was caught by a check or my own review, before anyone acted on a bad number.

FIG // THE LEDGER · AS OF 2026-07-09

THE OUTCOME		THE WORK	
analysis runs on the record	23	backlog items shipped	58
company analyses across them	160+	eval log findings	25
check families grading each run	8	fixed	13
blocking failures escaped	0	still open, one major	8
regression tests on its own codebase	281		

This page summarizes the system's eval log. The full log, timestamps included, is available on request.

7 weeks FROM FIRST COMMIT TO EVERYTHING ON THIS PAGE, BUILT NIGHTS AND WEEKENDS

THE PRODUCT CALL *A demo shows what an AI product can do.* **An operating record shows how it is managed.**

COHENEMC@GMAIL.COM

I built my career shipping product where every number faces an audit. *This system holds AI to the same standard.*

I scoped agentic AI strategy for Goldman Sachs' KYC program. Then I built and operate my own agentic system end to end, to know what that strategy feels like in practice.

That pilot carried two design signatures: a human at the decision point, and an agent that writes down its rationale for review. The system you just read runs on the same pattern.

Advising a bank showed me where the line belongs. Building one myself taught me what holding that line costs.

FORMAL TRAINING agentic AI systems, including the tooling this one is built on · AI evaluation · machine learning fundamentals (Stanford/DeepLearning.AI) · full credentials ↗

WHAT I'D BRING *The product calls an AI team needs made: where the human sits, what gets gated, how trust gets proven. I've made each one myself, so I know what I'm asking of* **the people who build it.**