honeycomb.io

30 to 70 PRs a day:
How we managed to not wreck our systems

May/June 2026

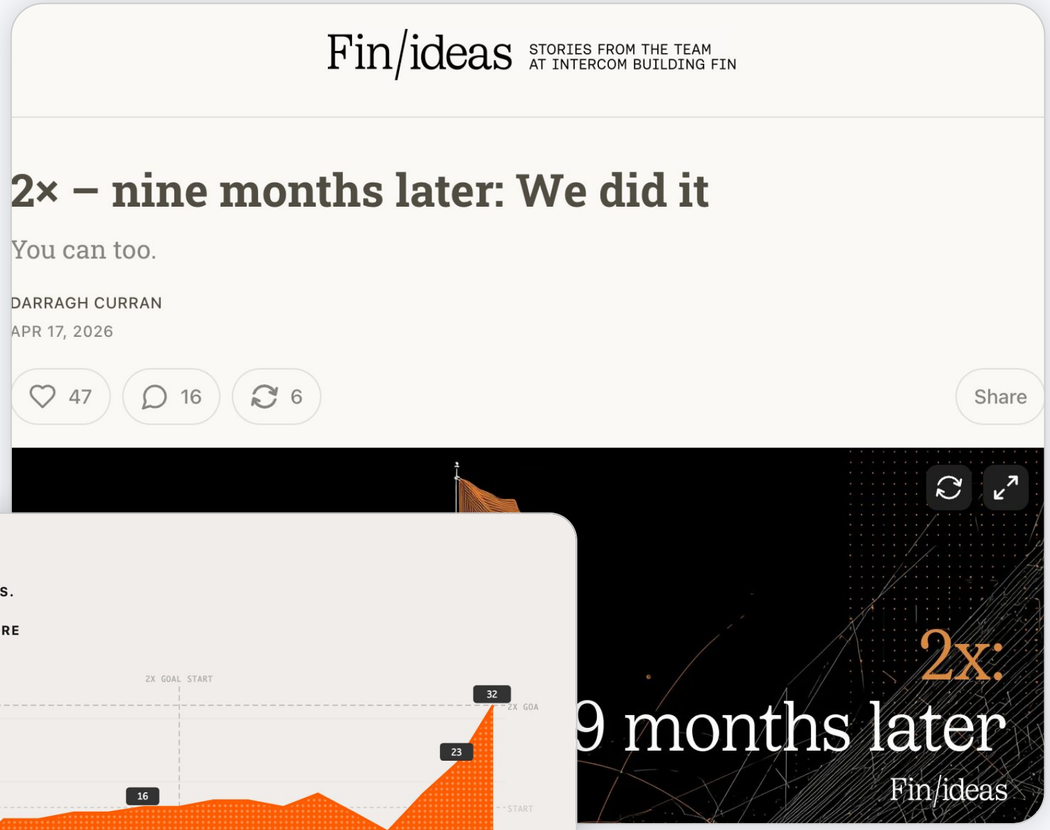
Liz Fong-Jones

Technical Fellow
Honeycomb

Intercom/Fin set the bar. We're following.

- **We credit them. We follow them.**
  Our orgs share more than they differ
- **They set a 2x goal, not a 10x goal. And hit it.**
  We wanted some of that
- **Transparent about methodology**
  and the required org-shape changes
- **But we have work to do to match their auto-approval (and that's okay)**

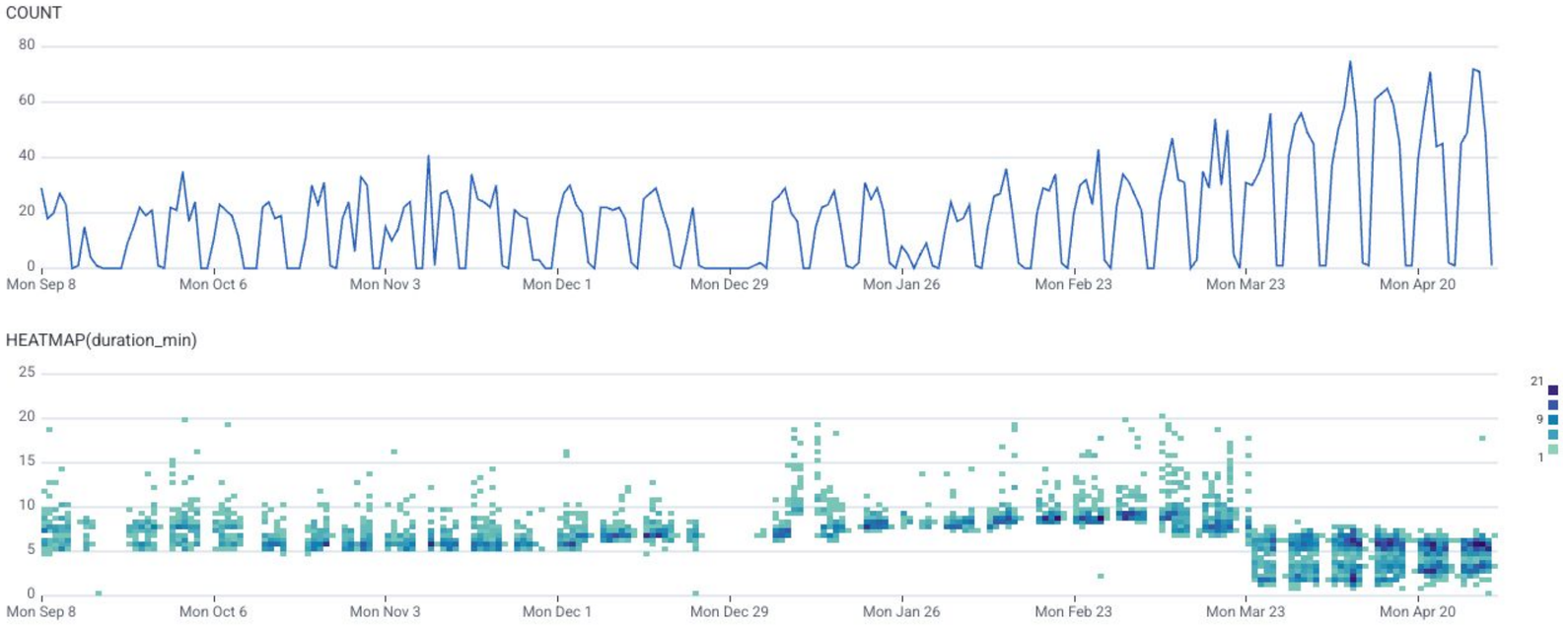
  Substrate first for confidence



https://ideas.fin.ai/p/2x-nine-months-later

Let’s be honest about that number

- **It’s peak weekday, not calendar average**
  Wednesday no-meeting days are the most productive, before & after
- **It’s a floor, not a true count**
  One of our heaviest Claude Code users has zero AI-attributed git commits
- **It's entangled with org changes happening at the same time**
- **Take every number you hear with a grain of salt. Including from me.**

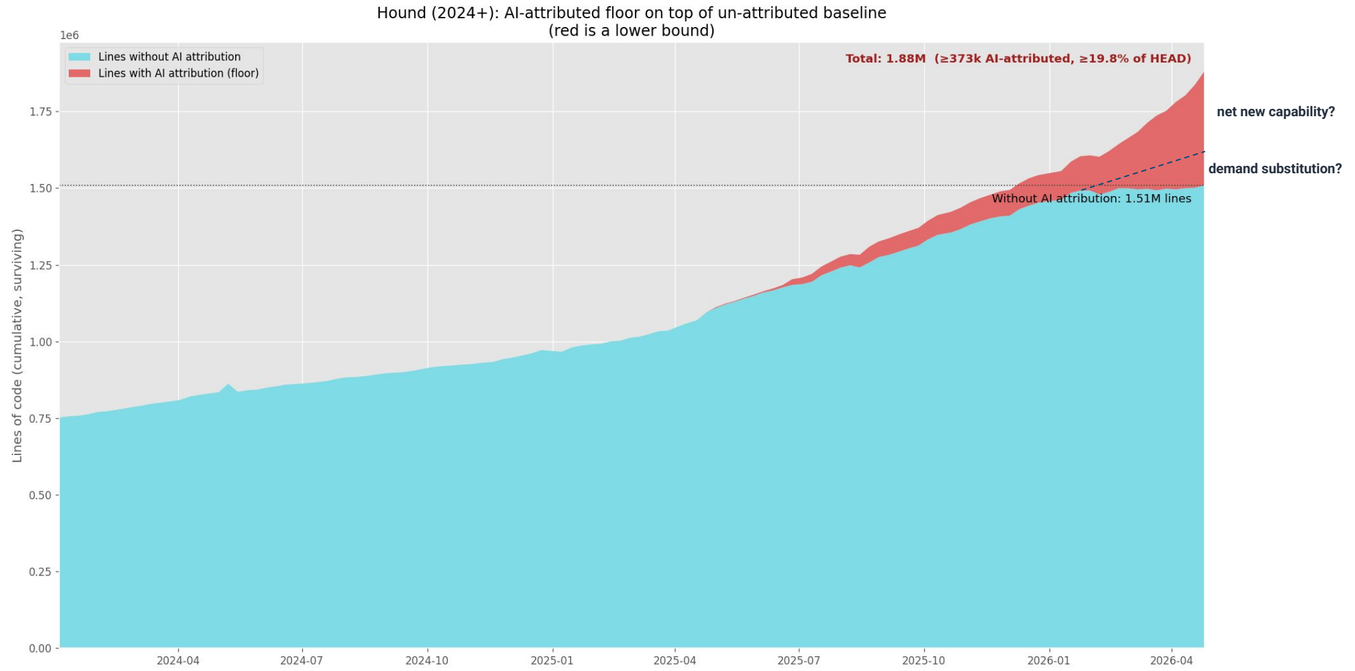


Peak weekday merges

~30 to ~74

- **Non-AI-attributed merges held steady at 25-30**
  Humans didn't slow down to make room
- **The +44 delta is from AI-attributed commits**
  Some new capacity, some substitution

  Can’t separate without a controlled experiment we don’t have



AI contribution isn't a binary

- **The trailer is binary; contribution is continuous**

  “AI was somewhere in this PR's history” is weaker than
  “AI wrote this.”
- **Two failure modes: squash-stripping AND trailers disabled**

  ~2.7× undercount, or ∞× — heaviest users may sit at zero git AI%
- **Floors come from git, ceilings from telemetry**

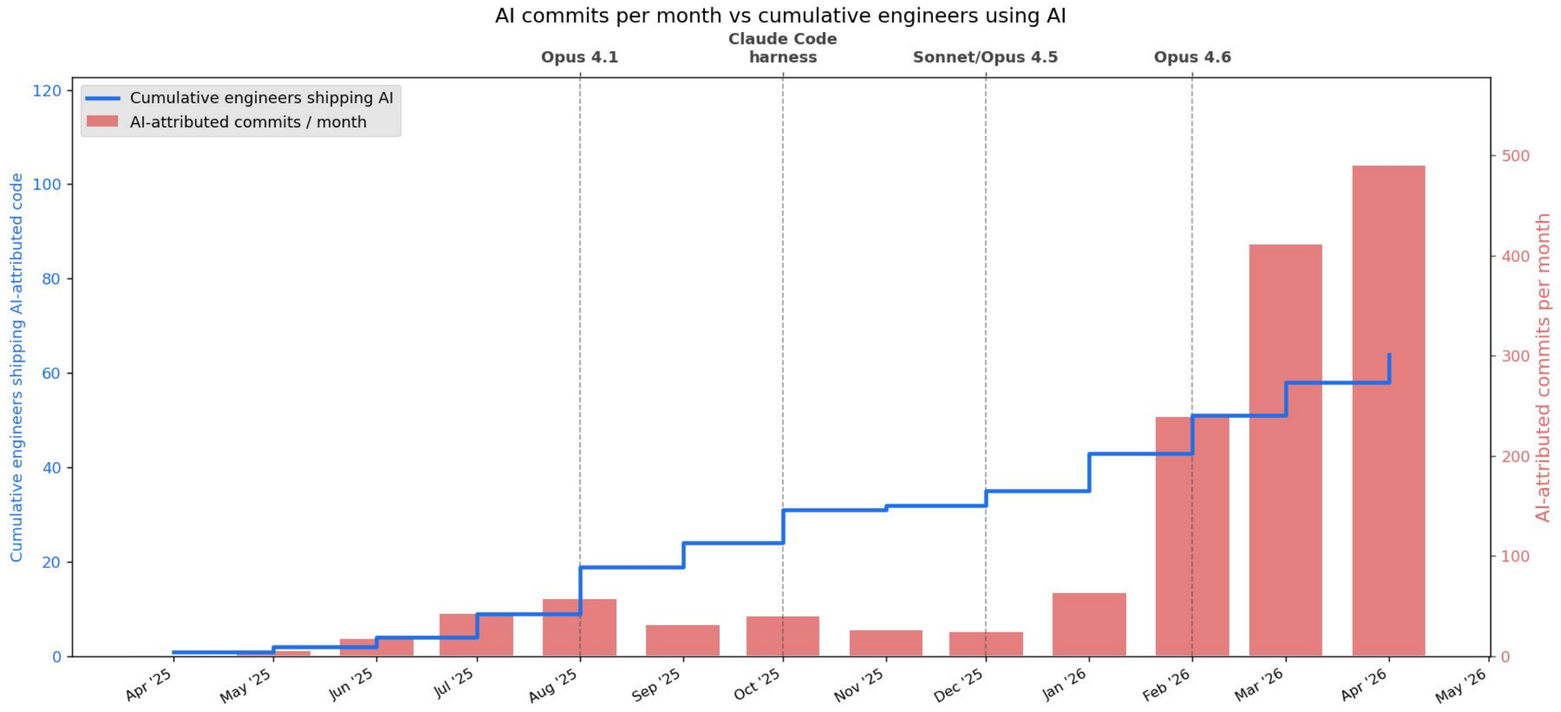
  ≥63% PRs / ≥75% lines (calibrated). ~85% of org ever-used. Different questions, not the same number

low-confidence: lol

high-confidence: verifiable

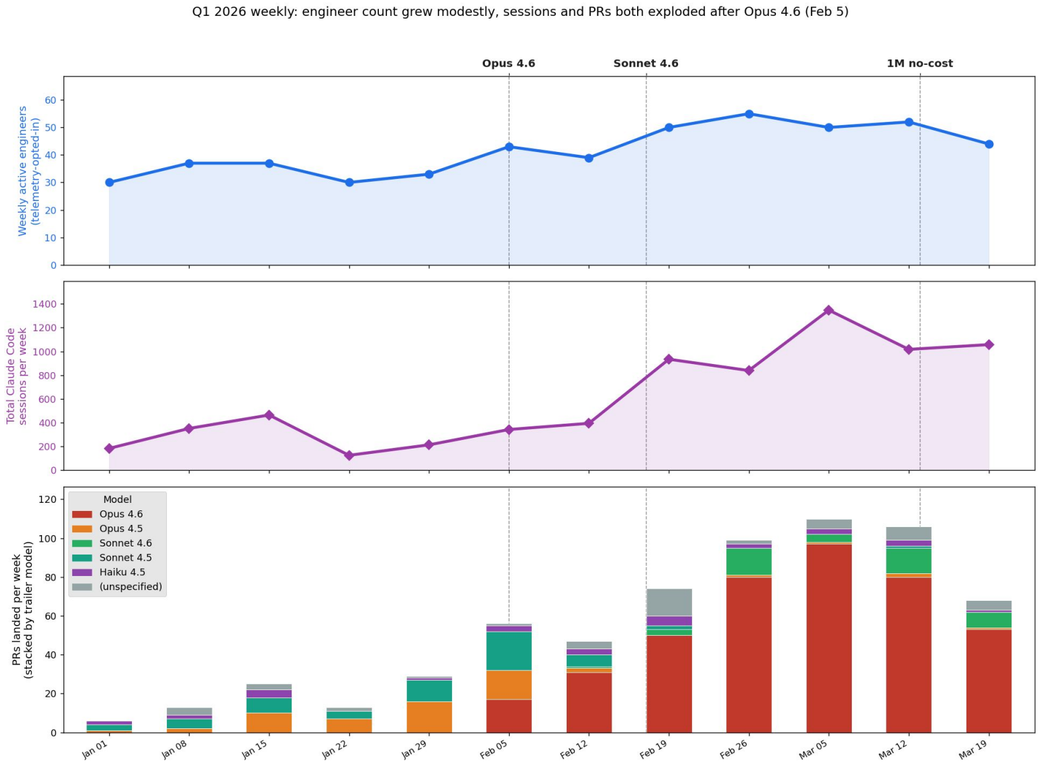
Leadership opened the door, we went through

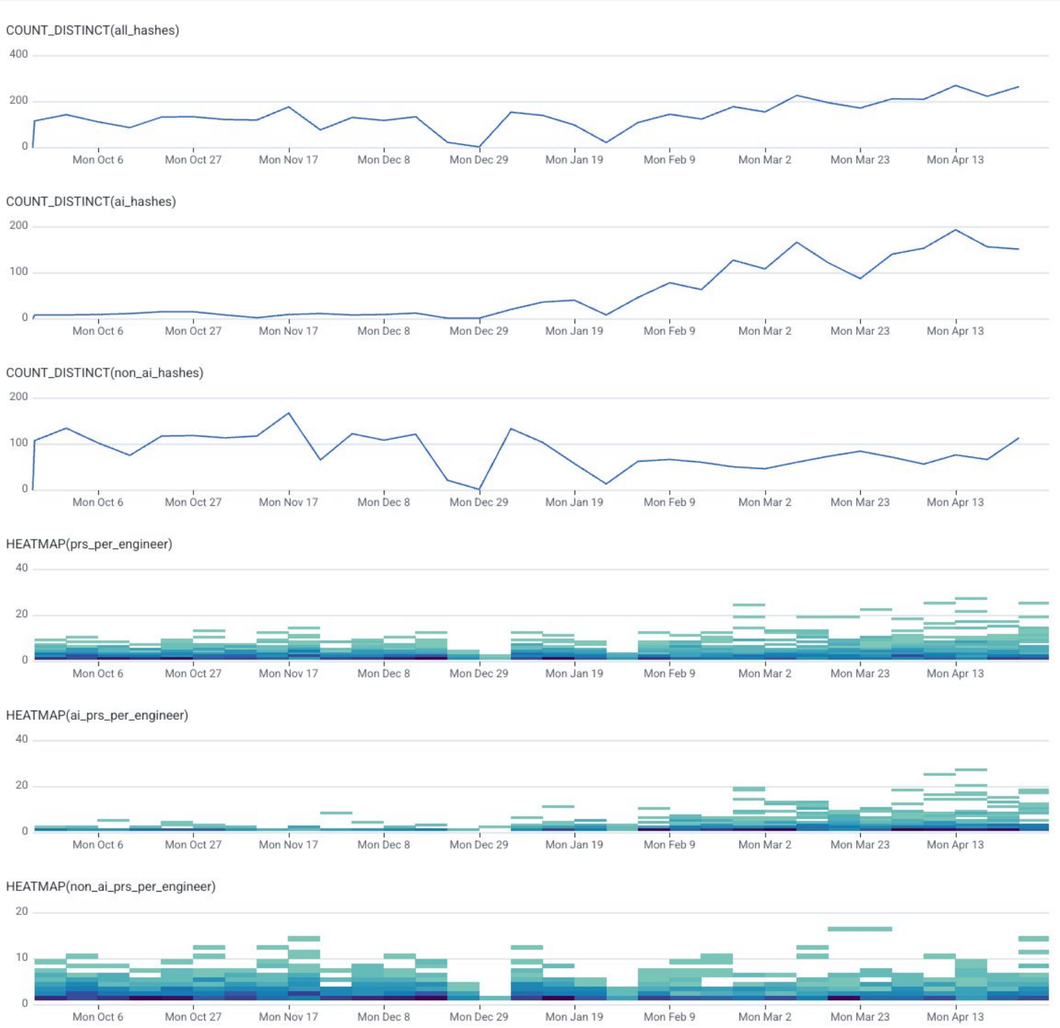
- **Aug 2025 (+10)** Leadership "room to experiment" + Opus 4.1
- **Oct 2025 (+7)** Claude Code harness improvements
- **Mid–Jan 2026** Realignment toward greenfield, higher-AI-leverage work
- **Feb 2026 (+9)** Opus 4.6 crossed delegation threshold (and 4.7 is even better)



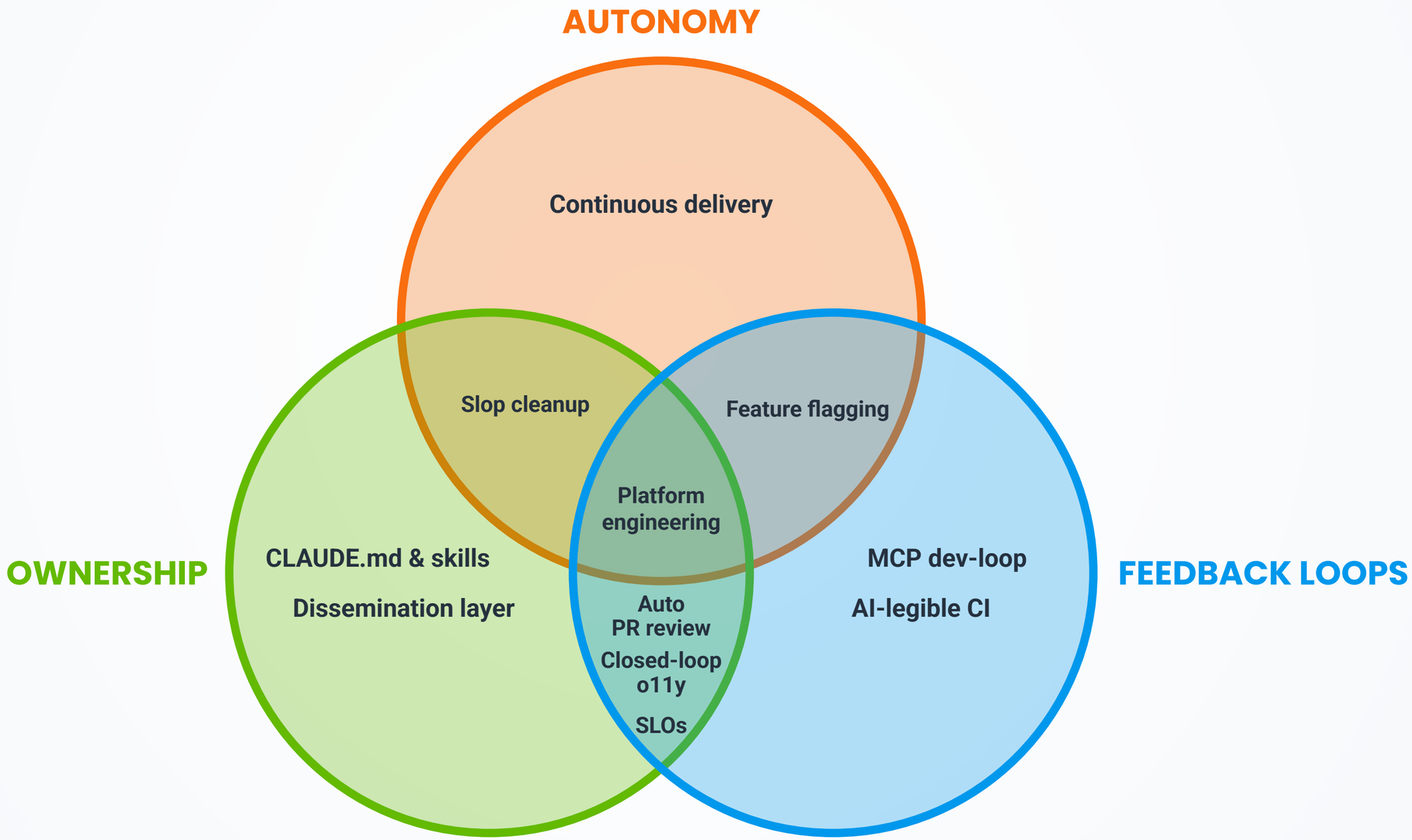
Feb 2026: enablement → delegation

- **Distinct users flat** 59 / 63 / 64 (Jan/Feb/Mar 2026)
- **3.3× sessions per user** 21 / 35 / 70 Same window
- **Same engineers, same tooling, new intensity**
  Lift is per-user-intensity, not headcount enabled





Autonomy, ownership, feedback loops



Continuous delivery: hourly deploy train

- **~Hourly train. 12-14 deploys/day. ~70 PRs. 1-3 reverts/reworks.**

  Recovery within an hour because the next train is right behind

- **Low latency deploys bound incident duration**

  Without this, the agent feedback loop doesn't close

- **Feature flags bound blast radius at the change level**

  Risky agent-shipped work gates behind flags; reverts and gradual rollouts both reduce change failure rate

- **Coming next: continuous-continuous deployments**

  Per-change soak times instead of batched trains.
  AI throughput exposed the next bottleneck.

RWX is not itself an AI tool. That's the point.

Feedback infrastructure has to be **AI-legible**, not AI-native

```
# Fetch CI Failure Logs
## Workflow
### Step 2a: RWX Logs

1. Get failing tasks:

```bash
rwx results <run-id>
```

If this returns an authentication error, **stop** and ask the user to run `rwx login`. Do not retry.

Identify the failing task(s) and their task IDs.

2. Download logs:

```bash
rwx logs <task-id>
```

This downloads log files to `.rwx/downloads/`.

3. Read the downloaded log file with the Read tool to understand the failure.
```

Fast AND AI-legible CI

- **Fast**

  AI shortens the writing loop; the testing loop has to shorten with it. 60-min coding + 15-min CI = fine. 5-min AI change + 15-min CI = build dominates.

- **AI-legible**

  CLI readable by agents (with skill files). Agents run code analysis and fix their own build failures via the same CLI a human would

- **Not direct causation for speedup, but how we didn't fall over**

  Shipping that many changes in parallel would have become impossible without Namespace and RWX

- **Enables other tooling**

  RWX hosts our Claude Code auto-review agents too

CLAUDE.md and skills: ownership reframed

- **Authorship is no longer load-bearing**

  Humans are accountable for what they allow AI to ship

- **CLAUDE.md and skills make team standards legible to the substrate writing the code**
- **The accountability flywheel**

  Misses become substrate updates.
  Not “we're on the hook”; “we close the loop when we miss.”

- **Platform accountable for the system as a whole**

  Regardless of skill of the tool user

Auto PR review, empowered humans push back

- **Humans empowered to push back is load-bearing**

  Sometimes vigorously. It's how you enforce norms.

- **#32989 (closed unmerged, simpler PR merged):**

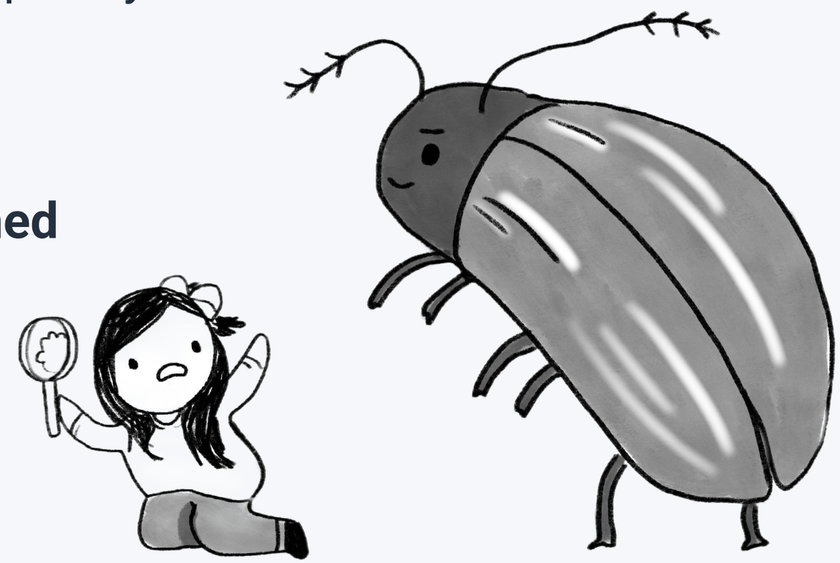
  Ian caught auto-review bot's hallucinated race condition.
  “I would be fine with the original, slower implementation.”
  Simpler version (#33045) shipped: 80% of gain at 10% of complexity

  #33124 (merged after rework): three-pass discipline
  Broad critique → “Obvious AI slop” → “ok let's try it”

- **A senior engineer’s detailed attention is allocated, not assumed**

  And the practice has to reproduce all the way down

- **Just because you CAN with AI doesn't mean you SHOULD**



MCP: the tools the human had

- **Same-session ticket → fix → PR**

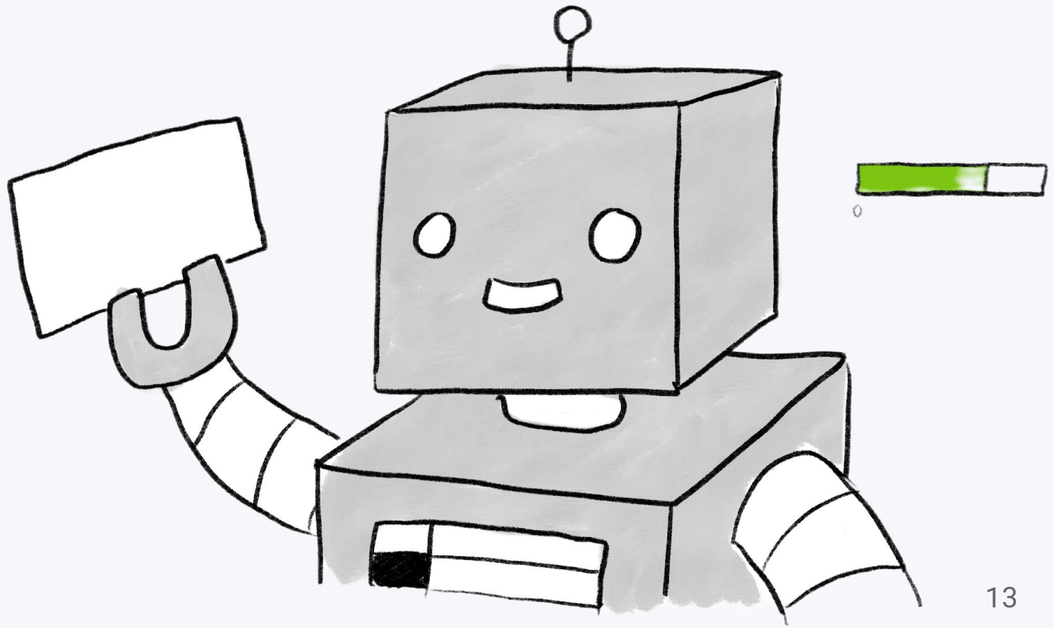
  In one Claude session, file a Linear ticket, fix, submit a PR pointing at it. Intent capture happens automatically.

- **The cost of intent capture went to near-zero**

  Not that it stops mattering

- **Honeycomb MCP** | Production telemetry
- **Playwright MCP** | UI verification
- **Notion MCP** | Design docs, RFCs
- **Linear MCP** | Tickets, user stories, acceptance criteria
- **An agent without these is flying blind**

  A junior wouldn’t be



Closed-loop observability for agent work

1. **Instrument the shipped code, not just the agent**

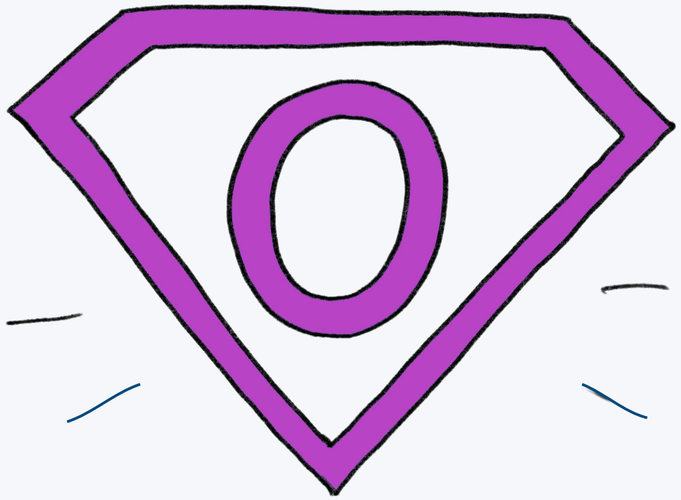
   Span attributes that link the running code back to the PR and agent session that created it. Not “tokens spent”; rather “this PR shipped Tuesday is the source of Wednesday's anomaly.”

2. **Production outcomes matter more than agent activity**

   Latency, error rate, cost, user behaviour on agent-shipped features. Did it work? What happened to real users?

3. **Closed-loop: production signal feeds the next change**

   Cost-per-interaction SLOs that fire back into agent context. Anomalies in BubbleUp auto-converting into review-agent rules.



AI for slop cleanup, not just generation

- **Deletion is productivity**

  Migrations. Dead-code. Dependency upgrades. Doc rewrites.

- **Capacity allocation is a choice**

  Reinvest in the substrate to sustain more, not in marginal features

- **AI lets you do the side quests**

  Things we ought to do that deliver quick wins, but used to cost a context switch. Dependency merges: an agent has time to read the changelog and run the tests.

- **Reclaiming work humans were doing badly is reallocation, not surrender of autonomy**

The dissemination layer

Code review does two jobs: catch bugs AND produce shared understanding

Automate the first; the second needs its own surface area, decoupled from approval gating.

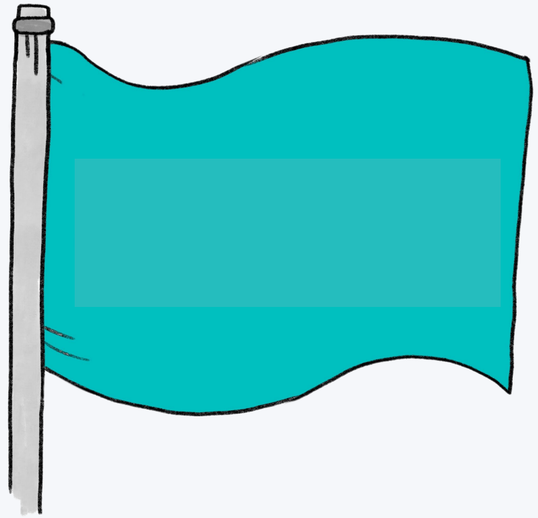
Honeycomb in production: Argo, Deploy Train, Terraform Hush, and Honeycomb Private Cloud's changelog watch

All built with AI. We used AI capacity to build the tooling that lets humans stay aware of what AI is shipping.

Two more practices to contain risk

Feature flagging

Agents can be given the ability to gate their own changes for gradual rollout. The flag itself becomes the feedback signal.



SLOs

AI features need SLOs on cost-per-interaction and latency, not just availability. An ownership statement AND a feedback mechanism.

The metric I’d rather put on the wall

- **Not PRs/day**

  Throughput is an input metric, not the product.

  But it's the coarse measure we have today for step change.

- **Rework rate over 6 months**

  Where speed has to survive maintenance load

- **Sustained-use metrics, not first-week clicks**

  UX score, dead clicks, task success, sentiment.
  Where the user outcome shows up.

Where the sceptics are right (two concessions)

Bank-LOB engineering won't transfer

"You are not going to radically realign LOB engineering at a bank."

What makes this work at Honeycomb is downstream of cultural conditions that don't exist at a regulated enterprise

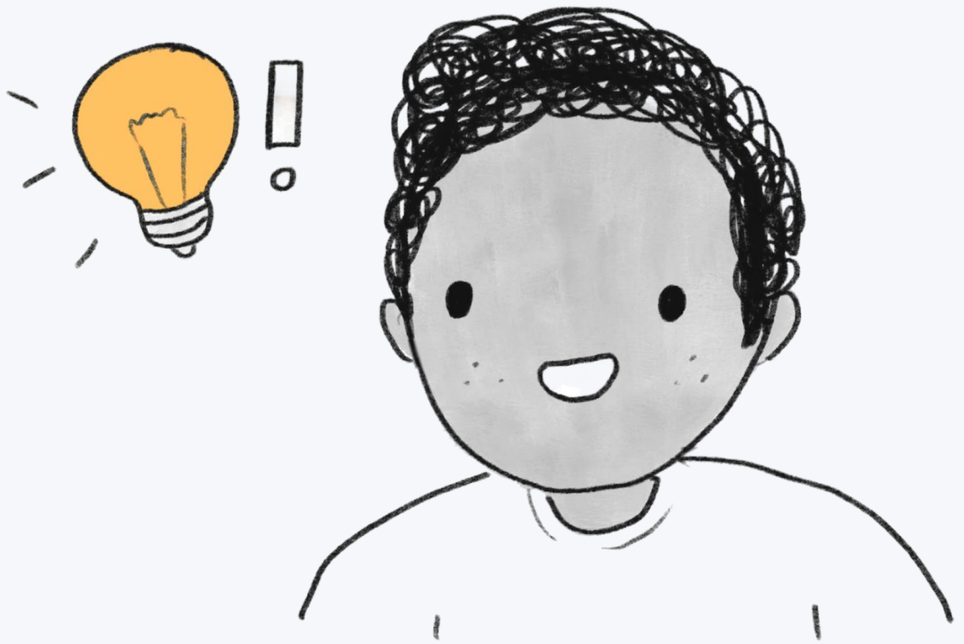
Spot-fix vs systematic-fix; complexity has a cost

AI makes spot-fixes so cheap that you keep reaching for them when systematic answers would be better

Write-cost dropping doesn't make maintenance-cost drop with it

Order of adoption

1. **CD + observability/SLOs as paired foundation**
   Each without the other is half a story
2. **Code ownership + CLAUDE.md and skills**
   Without these, AI amplifies confusion
3. **Hermetic, speedy builds + feature flags as you scale**
   Reproducibility for verification; gradual rollout for safety
4. **Chaos engineering as audit, not foundation**
   Once the rest is solid



Structuring the platform team

- **Classic platform work + maintaining the agent substrate**
  CLAUDE.md, MCP servers, AI-legible CIs, dissemination tooling, auto-review rules
- **Intercom's pattern: "team-2x" + plugin marketplace** (hundreds of contributors & skills)
  Modernising the factory is everyone’s job
- **The platform team maintains the agent's relationship to the platform**
  Not just the platform

Three things to take away

- **AI amplifies your existing practices**

  Going fast without autonomy, ownership, and feedback loops is enshittification

- **Be deliberate about capacity allocation**

  Reinvest the new throughput in the platform substrate that lets you absorb more, not in marginal features

- **“We didn't obviously break anything” is not success on its own**

  No one in this room is done.
  Including us.

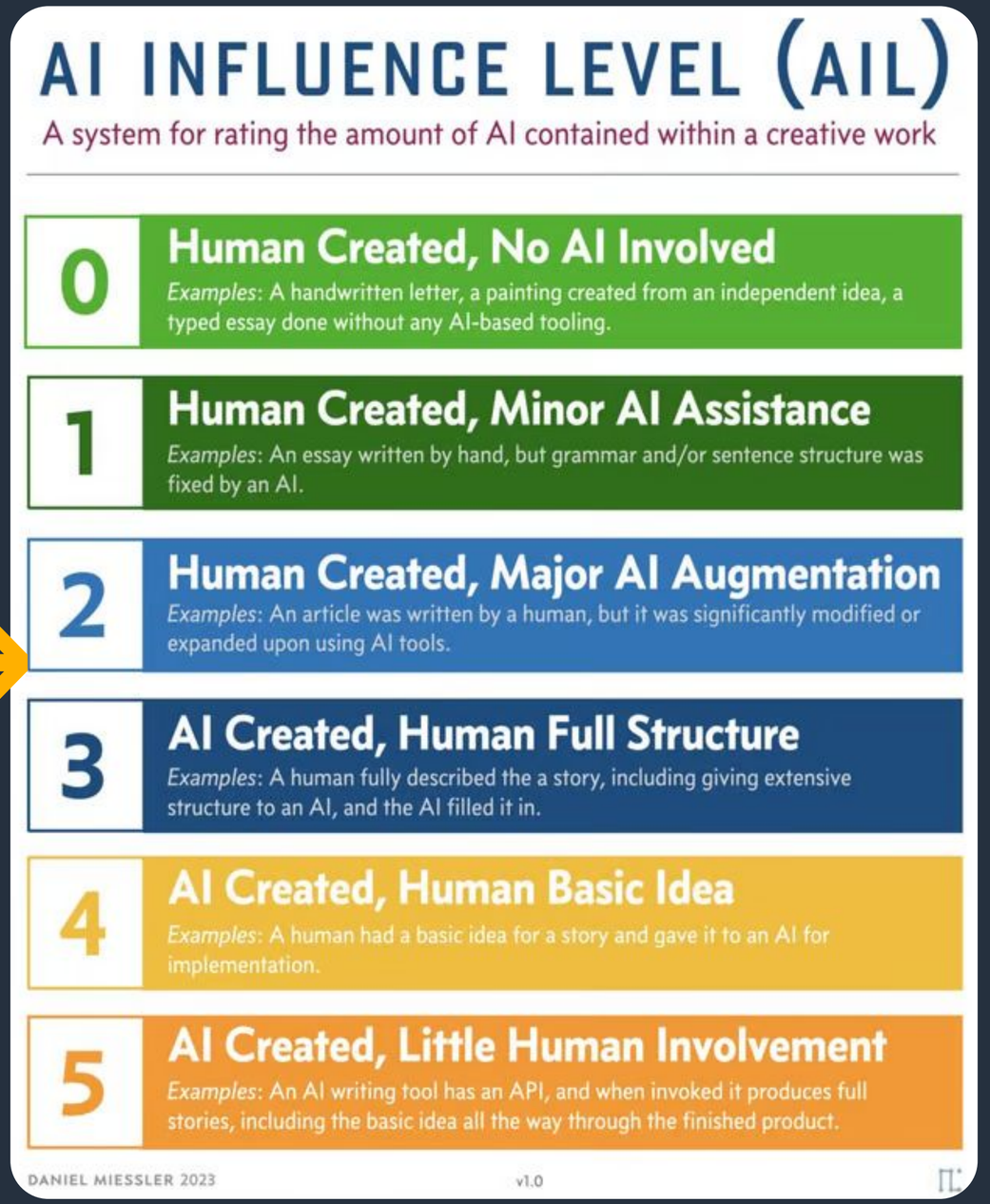
Thank you.

danielmiessler.com/blog/ai-influence-level-ail

Slide text & speaker notes: AIL-2.5

Images: AIL-0 (by bbghost.bsky.social)

Art should be made by artists, not machines



honeycomb.io

Liz Fong-Jones

Technical Fellow | Honeycomb

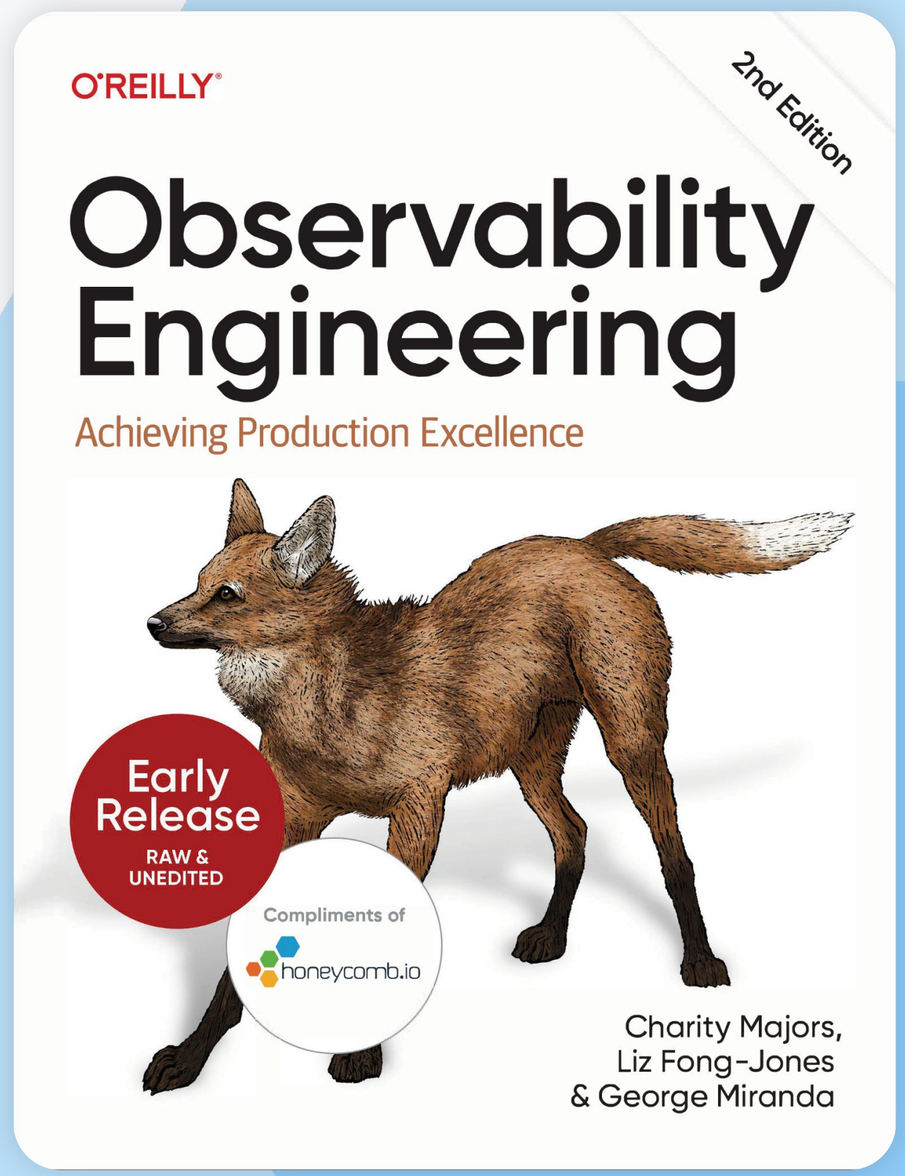
Sydney, NSW | MACS (Snr) CP

Vancouver, BC

linkedin.com/in/efong

lizf@honeycomb.io

@lizthegrey.com



honeycomb.io

30 to 70 PRs a day: How we managed to not wreck our systems

May/June 2026

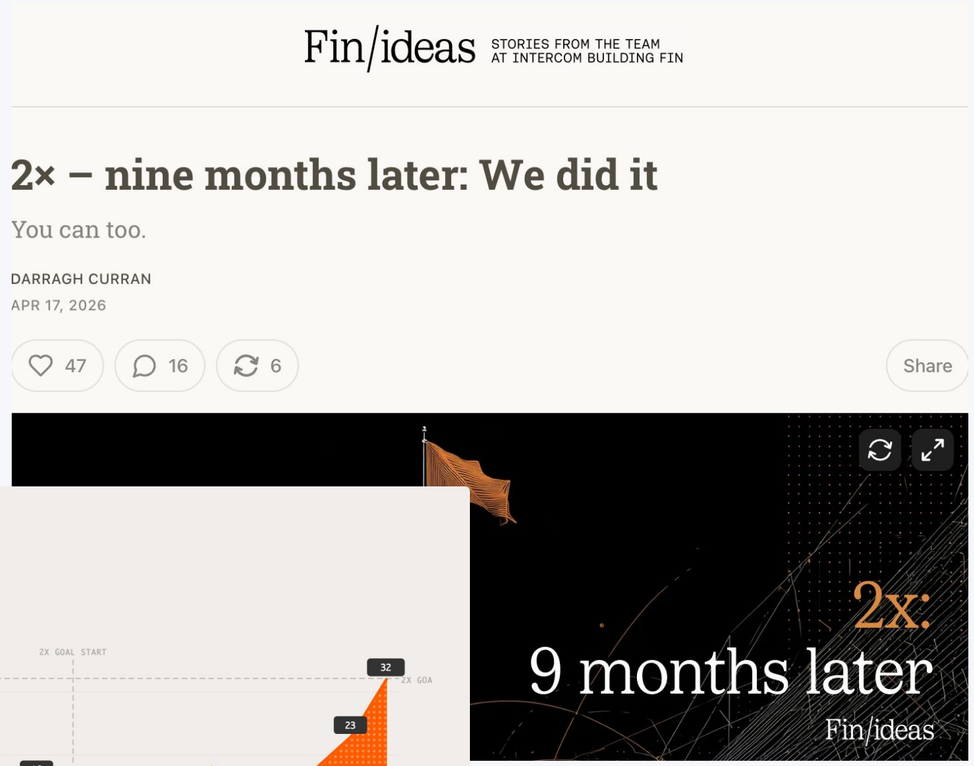
Liz Fong-Jones, Technical Fellow
Honeycomb

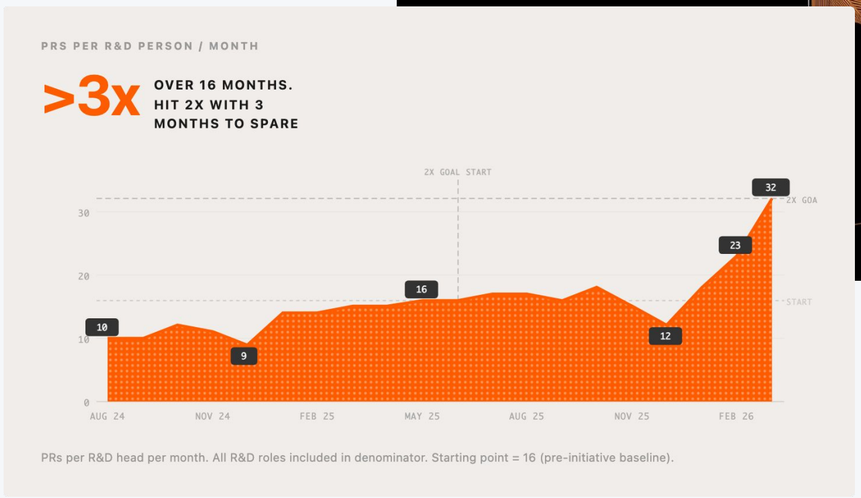
An optimist and two sceptics walk onto a meetup stage...

Intercom/Fin set the bar. We're following.

- **We credit them. We follow them.**
  Our orgs share more than they differ.
- **They set a 2x goal, not a 10x goal. And hit it.**
  We wanted some of that.
- **Transparent about methodology.**
  and the required org-shape changes.
- **Our zero auto-approval is a deliberate position (for now).**
  Substrate first for confidence.

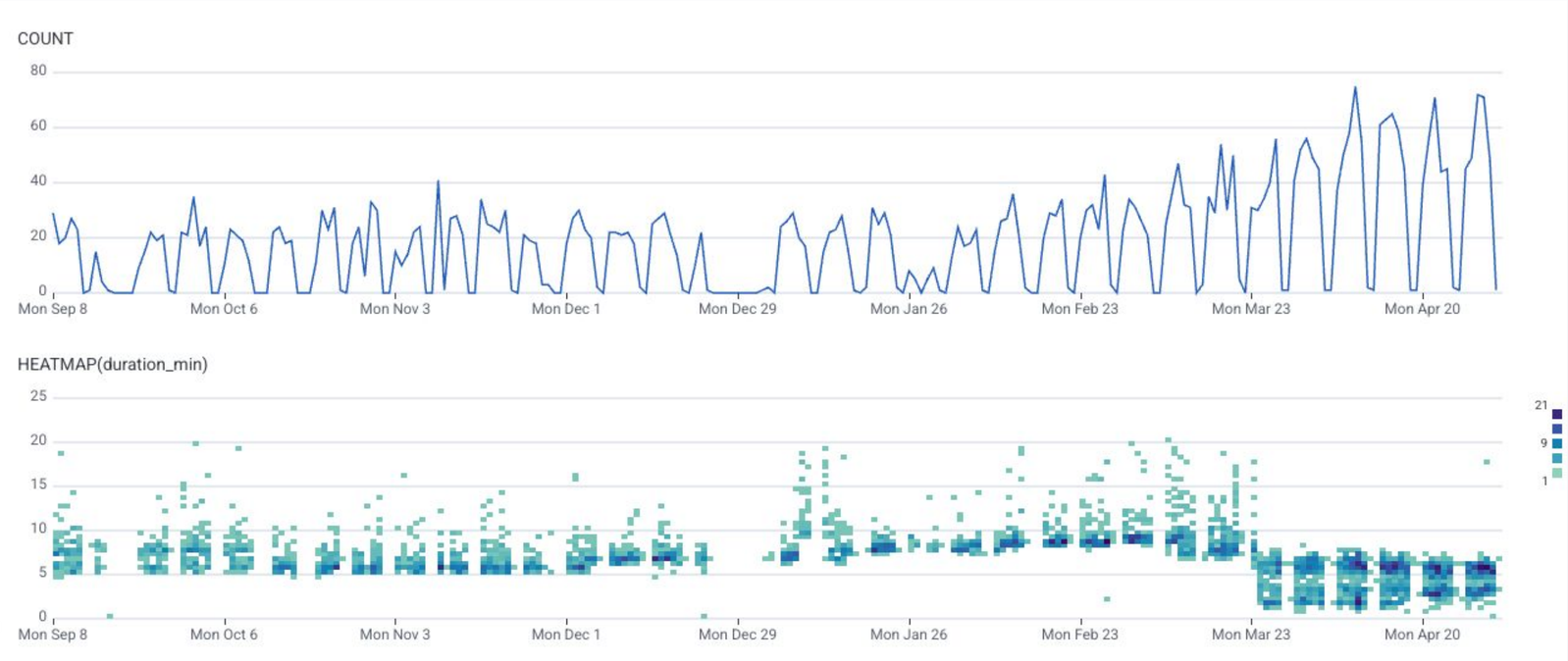
https://ideas.fin.ai/p/2x-nine-months-later





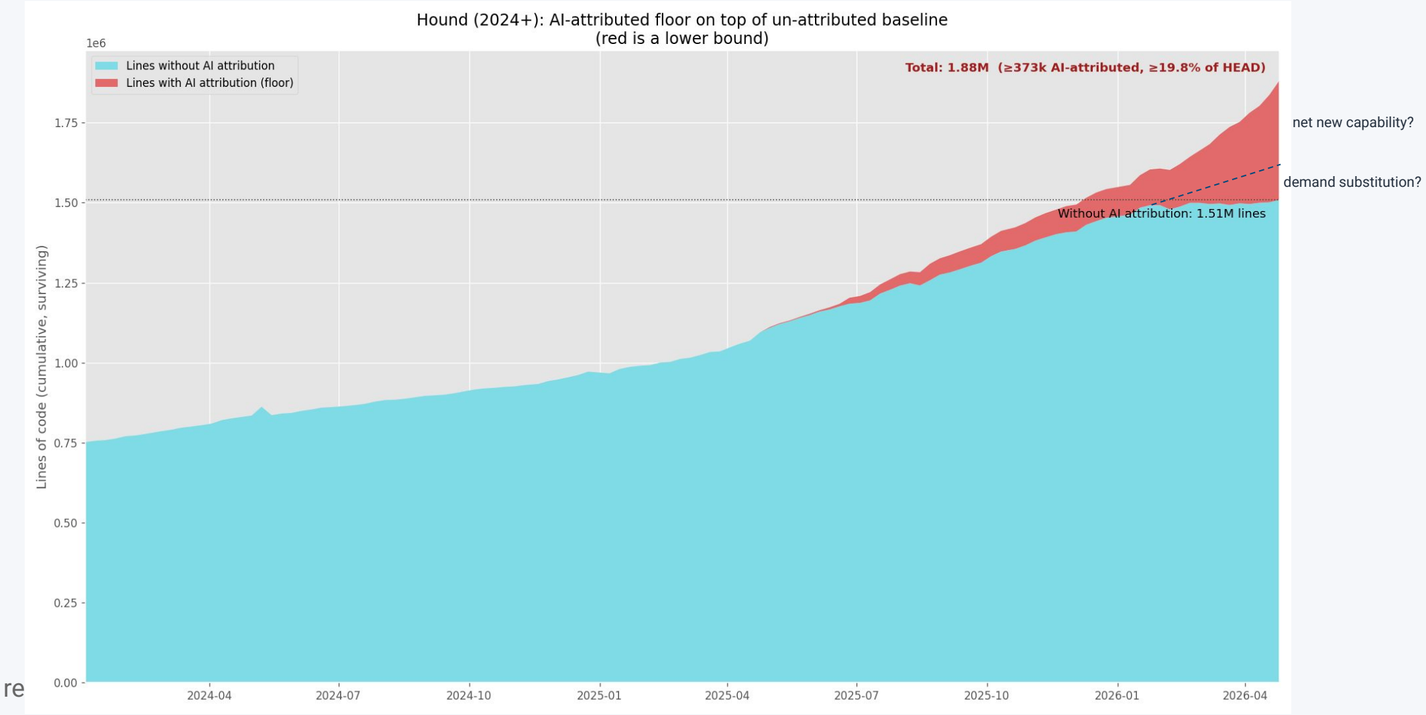
Let's be honest about that number

- **It's peak weekday, not calendar average.** Wednesday no-meeting days are the most productive, before & after.
- **It's a floor, not a true count.** One of our heaviest Claude Code users has zero AI-attributed git commits.
- **It's entangled with org changes happening at the same time.**
- **Take every number you hear with a grain of salt. Including from me.**



Peak weekday merges: ~30 to ~74

- **Non-AI-attributed merges held steady at 25-30.**
  Humans didn't slow down to make room.
- **The +44 delta is explicitly AI-attributed.**
  Some new capacity, some substitution.
  Can't separate without a controlled experiment we don't have.



AI contribution isn't a binary

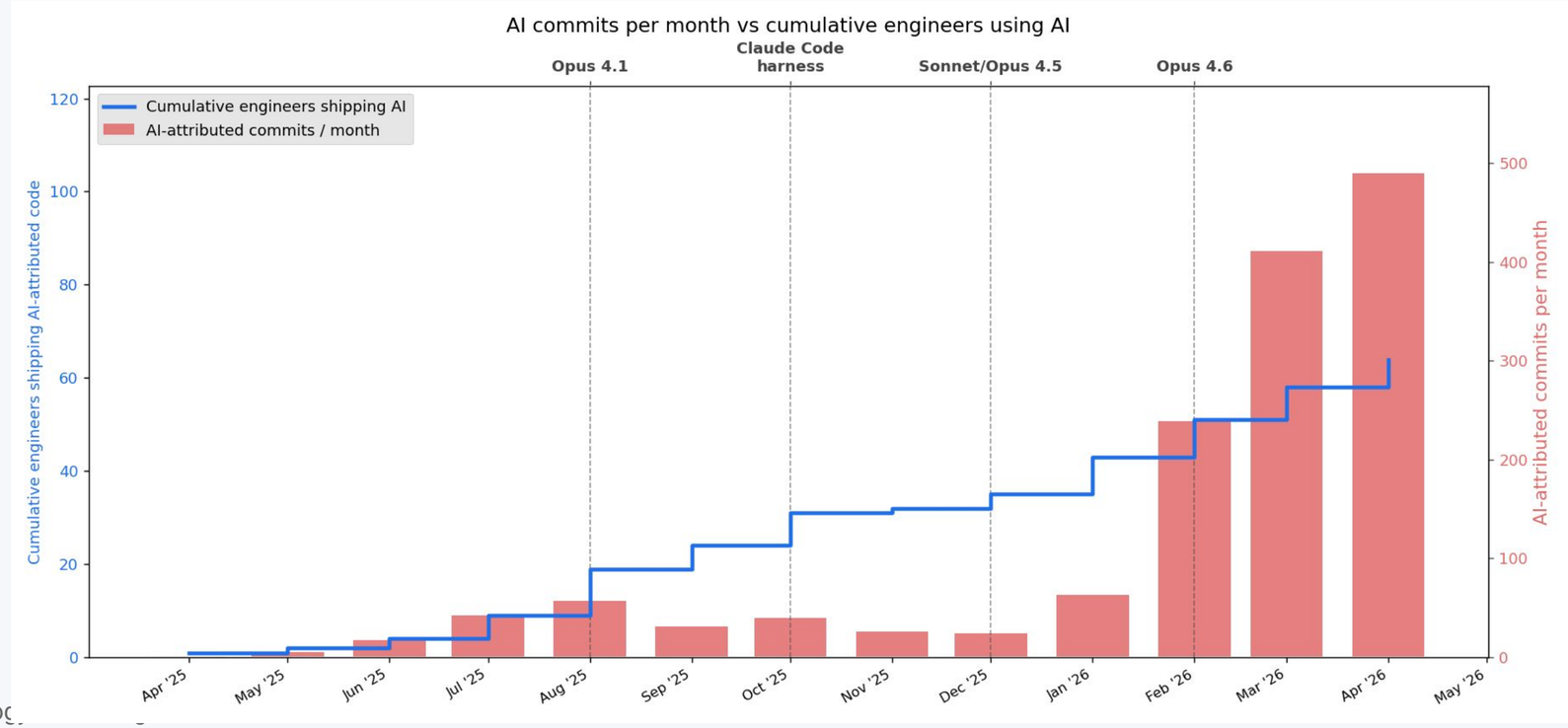
- **The trailer is binary; contribution is continuous.**
  "AI was somewhere in this PR's history" is weaker than "AI wrote this."
- **Two failure modes: squash-stripping AND trailers disabled.**
  ~2.7× undercount, or ∞× — heaviest users may sit at zero git AI%.
- **Floors come from git, ceilings from telemetry.**
  ≥63% PRs / ≥75% lines (calibrated). ~85% of org ever-used.
  Different questions, not the same number.

low-confidence: lol

high-confidence: verifiable

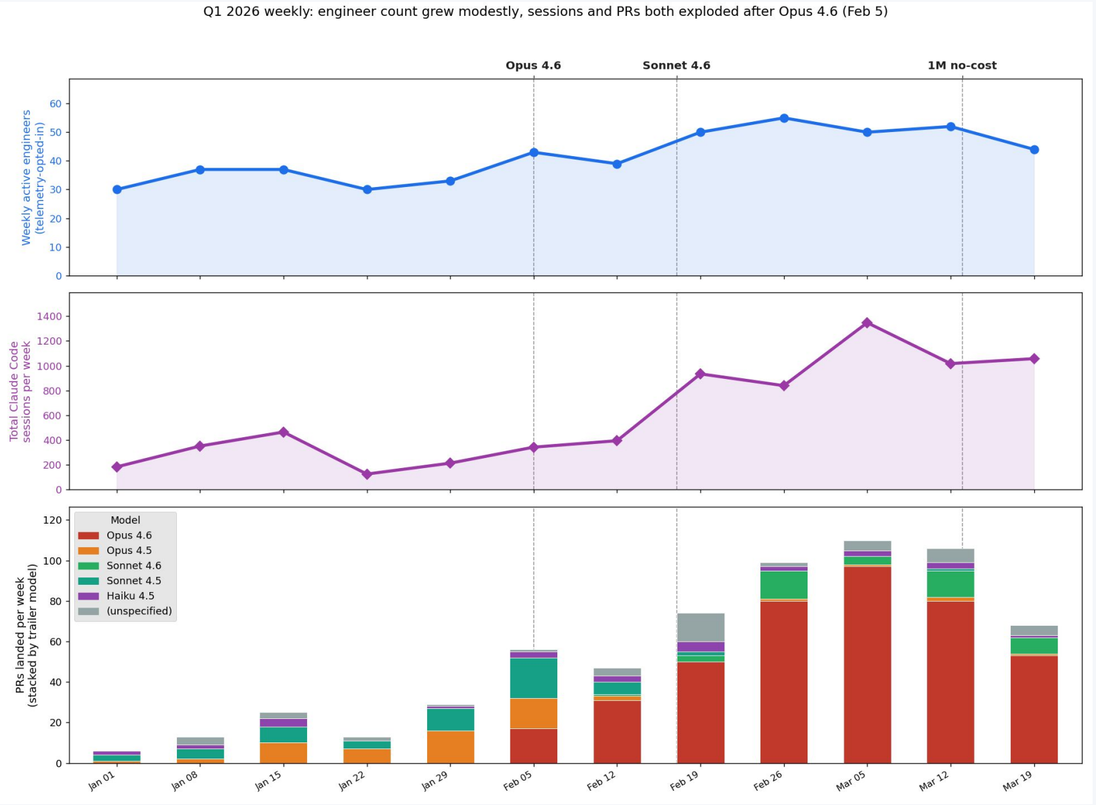
Leadership opened the door, we went through

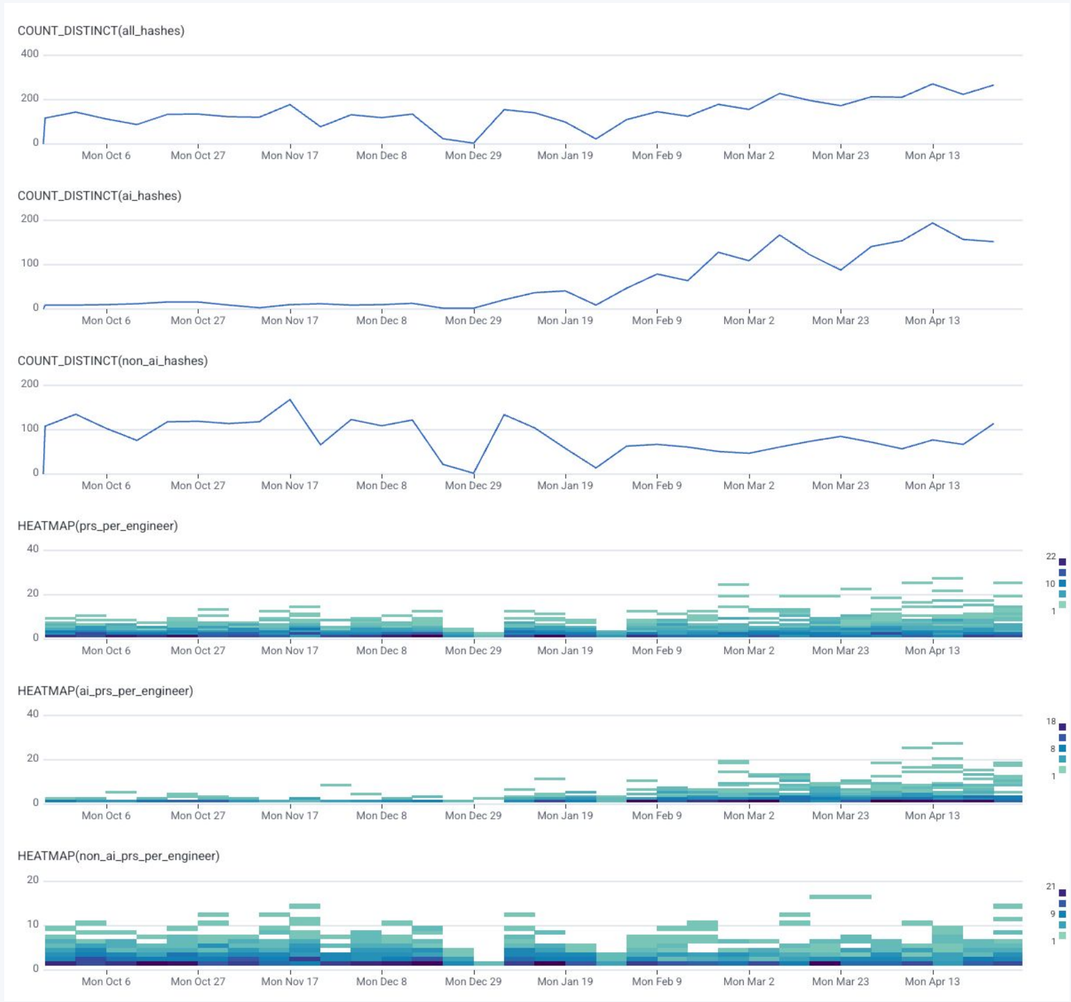
- **Aug 2025 (+10).** Leadership "room to experiment" + Opus 4.1.
- **Oct 2025 (+7).** Claude Code harness improvements.
- **Mid-Jan 2026.** Realignment toward greenfield, higher-AI-leverage work.
- **Feb 2026 (+9).** Opus 4.6 crossed delegation threshold (and 4.7 is even better).



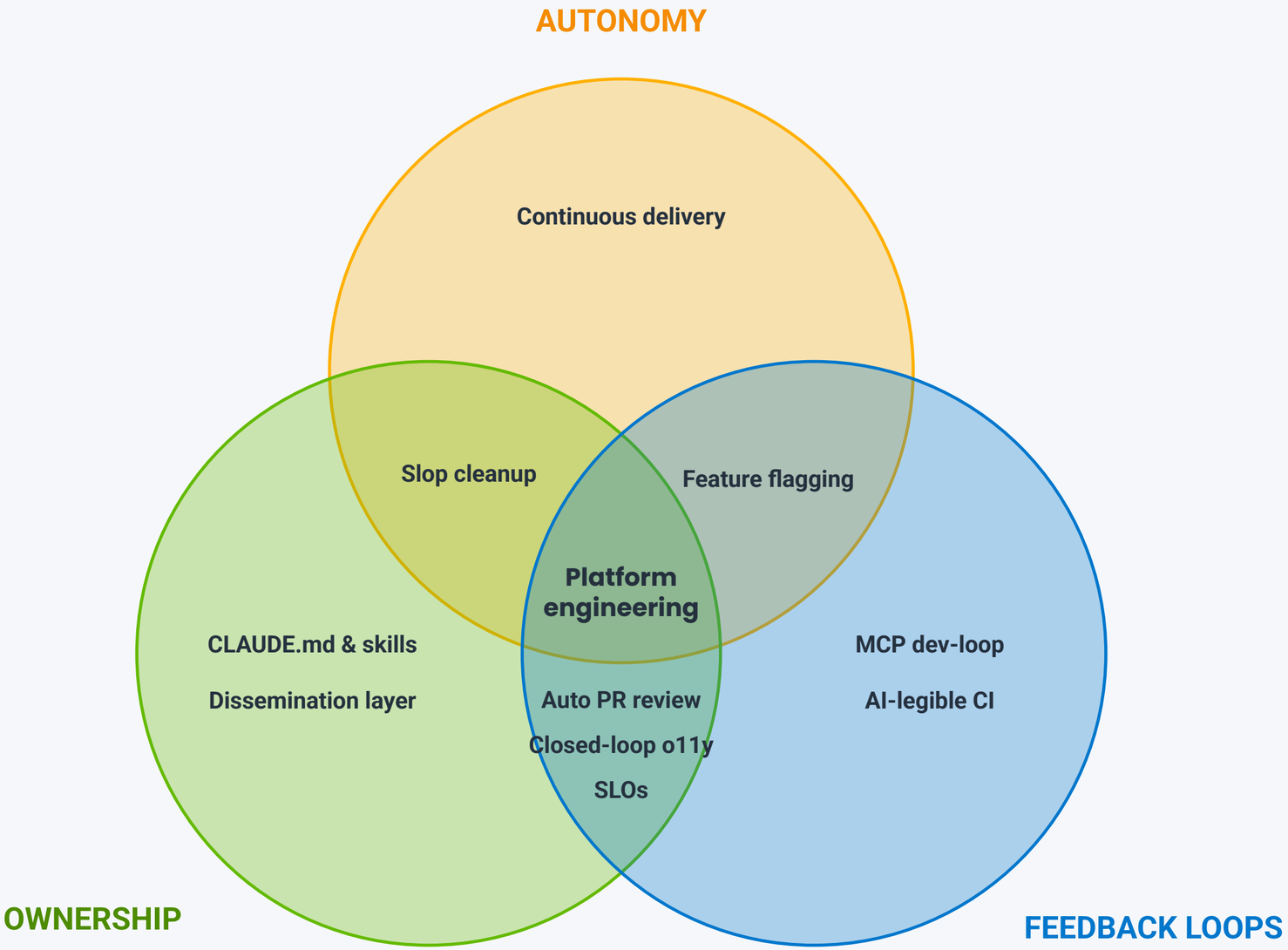
Feb 2026: enablement → delegation

- **Distinct users flat:** 59 / 63 / 64. (Jan/Feb/Mar 2026.)
- **3.3× sessions per user:** 21 / 35 / 70. Same window.
- **Same engineers, same tooling, new intensity.**
  Lift is per-user-intensity, not headcount enabled.



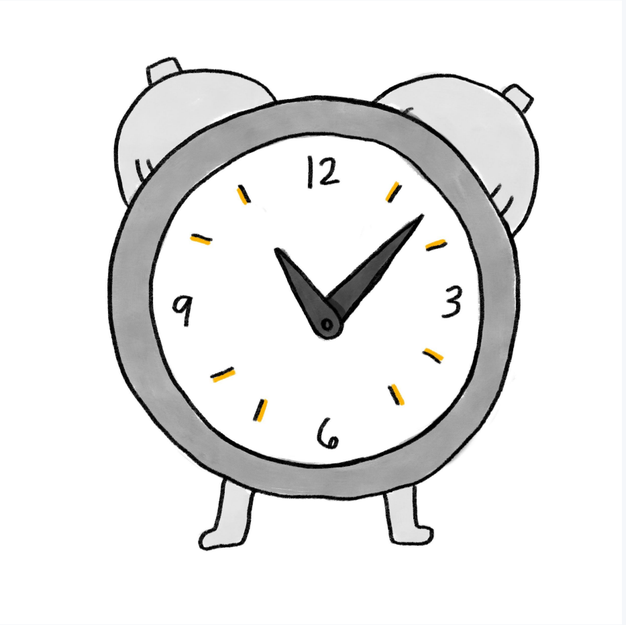


Autonomy, ownership, feedback loops



Continuous delivery: hourly deploy train

- **~Hourly train. 12-14 deploys/day. ~70 PRs. 1-3 reverts/reworks.**
  Recovery within an hour because the next train is right behind.
- **Low latency deploys bound incident duration.**
  Without this, the agent feedback loop doesn't close.
- **Feature flags bound blast radius at the change level.**
  Risky agent-shipped work gates behind flags; reverts and gradual rollouts both reduce change failure rate.
- **Coming next: continuous-continuous deployments.**
  Per-change soak times instead of batched trains.
  AI throughput exposed the next bottleneck.



Fast AND AI-legible CI

RWX is not itself an AI tool. That's the point. Feedback infrastructure has to be AI-legible, not AI-native.

```
# Fetch CI Failure Logs
## Workflow
### Step 2a: RWX Logs

1. Get failing tasks:

```bash
rwx results <run-id>
```

If this returns an authentication error, **stop** and ask the user to run `rwx login`. Do not retry.

Identify the failing task(s) and their task IDs.

2. Download logs:

```bash
rwx logs <task-id>
```

This downloads log files to `.rwx/downloads/`.

3. Read the downloaded log file with the Read tool to understand the failure.
```

Fast. AI shortens the writing loop; the testing loop has to shorten with it. 60-min coding + 15-min CI = fine. 5-min AI change + 15-min CI = build dominates.

AI-legible. CLI readable by agents (with skill files). Agents run code analysis and fix their own build failures via the same CLI a human would.

Not direct causation for speedup, but how we didn't fall over. Shipping that many changes in parallel would have become impossible without Namespace and RWX.

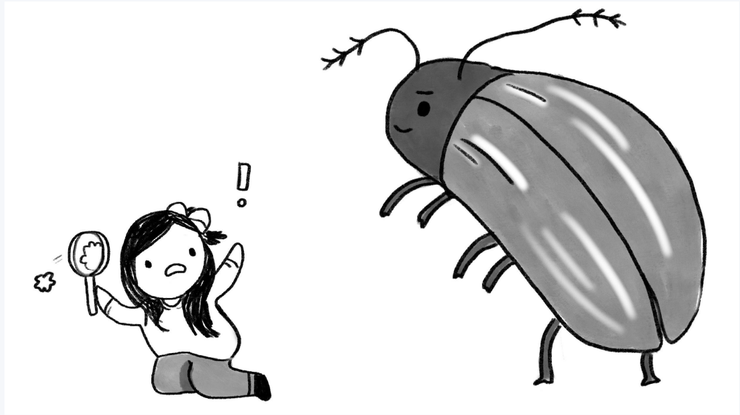
Enables other tooling. RWX hosts our Claude Code auto-review agents too.

CLAUDE.md and skills: ownership reframed

- **Authorship is no longer load-bearing.**
  Humans are accountable for what they allow AI to ship.
- **CLAUDE.md and skills make team standards legible to the substrate writing the code.**
- **The accountability flywheel.** Misses become substrate updates.
  Not "we're on the hook"; "we close the loop when we miss."
- **Platform accountable for the system as a whole.**
  Regardless of skill of the tool user.

Auto PR review, empowered humans push back

- **Humans empowered to push back is load-bearing.**
  Sometimes vigorously. It's how you enforce norms.
- **#32989 (closed unmerged, simpler PR merged):**
  Ian caught auto-review bot's hallucinated race condition.
  "I would be fine with the original, slower implementation."
  Simpler version (#33045) shipped: 80% of gain at 10% of complexity.
- **#33124 (merged after rework):** three-pass discipline.
  Broad critique → "Obvious AI slop" → "ok let's try it."
- **A senior engineer's detailed attention is allocated, not assumed.**
  And the practice has to reproduce all the way down.
- **Just because you CAN with AI doesn't mean you SHOULD.**



MCP: the tools the human had

Same-session ticket → fix → PR. In one Claude session, file a Linear ticket, fix, submit a PR pointing at it. Intent capture happens automatically.

The cost of intent capture went to near-zero. Not that it stops mattering.

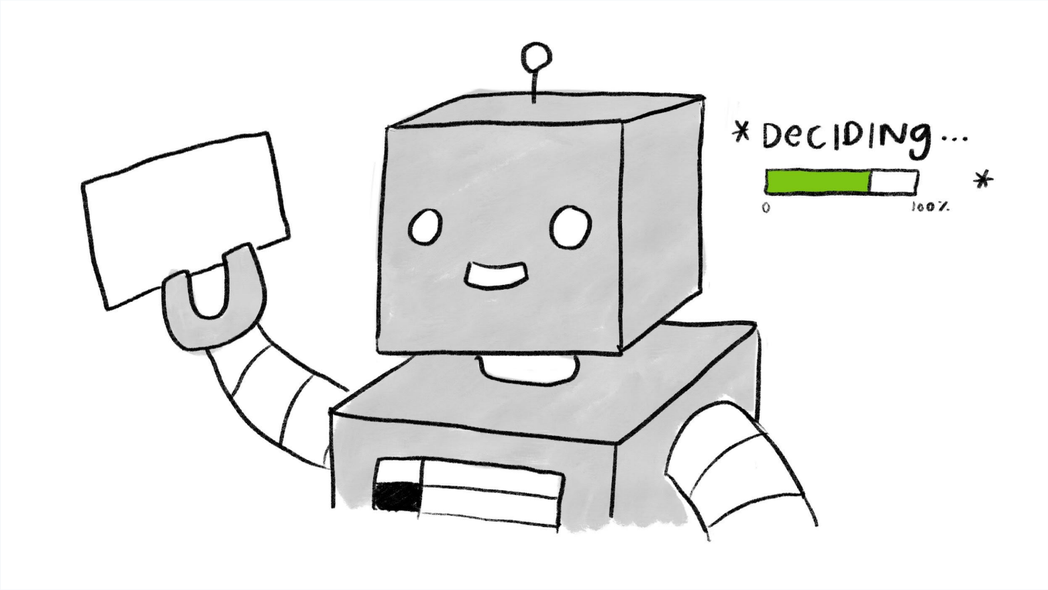
Honeycomb MCP: production telemetry.

Playwright MCP: UI verification.

Notion MCP: design docs, RFCs.

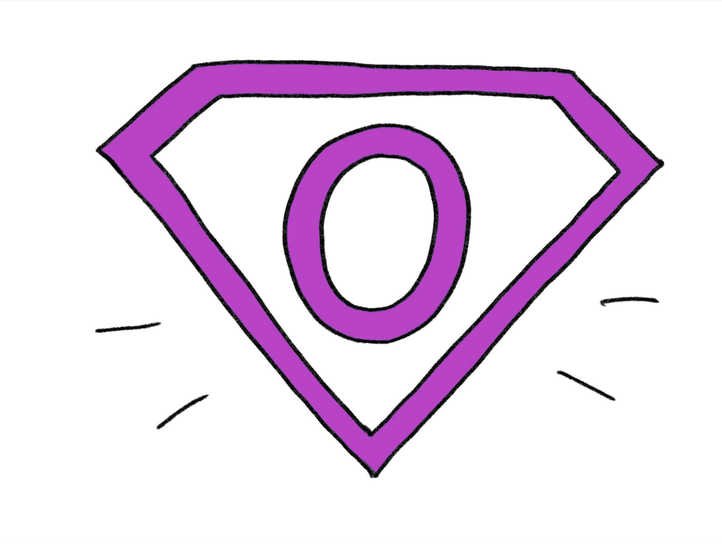
Linear MCP: tickets, user stories, acceptance criteria.

An agent without these is flying blind. A junior wouldn't be.



Closed-loop observability for agent work

1. **Instrument the shipped code, not just the agent.**
   Span attributes that link the running code back to the PR and agent session that created it. Not "tokens spent"; rather "this PR shipped Tuesday is the source of Wednesday's anomaly."
2. **Production outcomes matter more than agent activity.**
   Latency, error rate, cost, user behaviour on agent-shipped features. Did it work? What happened to real users?
3. **Closed-loop: production signal feeds the next change.**
   Cost-per-interaction SLOs that fire back into agent context. Anomalies in BubbleUp auto-converting into review-agent rules.



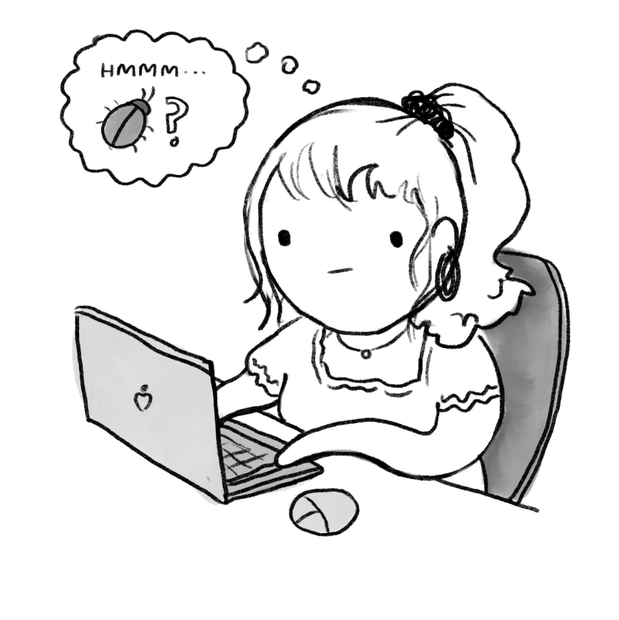
AI for slop cleanup, not just generation

- **Deletion is productivity.**
  Migrations. Dead-code. Dependency upgrades. Doc rewrites.
- **Capacity allocation is a choice.**
  Reinvest in the substrate to sustain more, *not* in marginal features.
- **AI lets you do the side quests.** Things we ought to do that deliver quick wins, but used to cost a context switch. Dependency merges: an agent has time to read the changelog and run the tests.
- **Reclaiming work humans were doing badly is reallocation, not surrender of autonomy.**

The dissemination layer

Code review does two jobs: catch bugs AND produce shared understanding.

Automate the first; the second needs its own surface area, decoupled from approval gating.

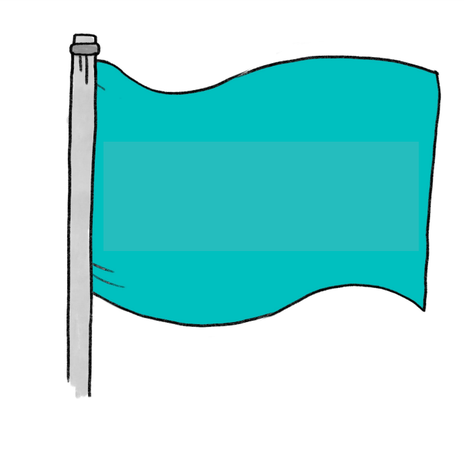


Honeycomb in production: Argo, Deploy Train, Terraform Hush, and Honeycomb Private Cloud's changelog watch.

All built with AI. We used AI capacity to build the tooling that lets humans stay aware of what AI is shipping.

Two more practices to contain risk

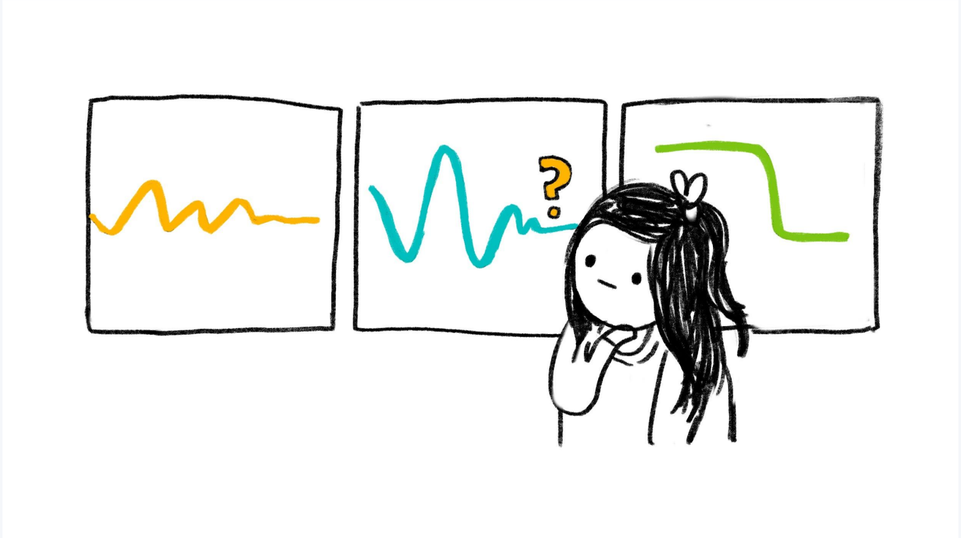
Feature flagging. Agents can be given the ability to gate their own changes for gradual rollout. The flag itself becomes the feedback signal.



SLOs. AI features need SLOs on cost-per-interaction and latency, not just availability. An ownership statement AND a feedback mechanism.

The metric I'd rather put on the wall

- **Not PRs/day.** Throughput is an input metric, not the product. But it's the coarse measure we have today to demonstrate step change.
- **Rework rate over 6 months.** Where speed has to survive maintenance load.
- **Sustained-use metrics, not first-week clicks.** UX score, dead clicks, task success, sentiment. Where the user outcome shows up.



Where the sceptics are right (two concessions)

Bank-LOB engineering won't transfer.

"You are not going to radically realign LOB engineering at a bank."

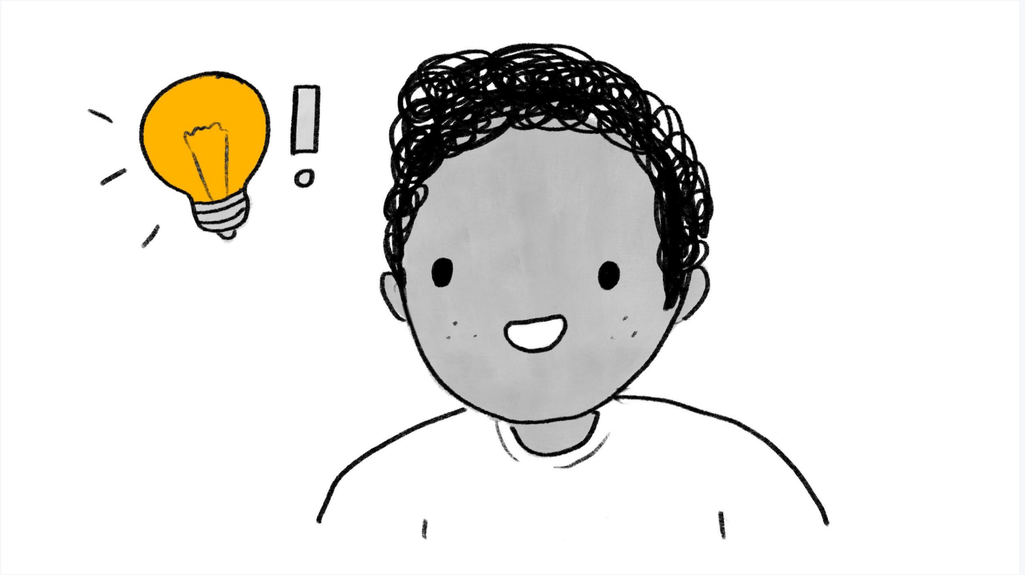
What makes this work at Honeycomb is downstream of cultural conditions that don't exist at a regulated enterprise.

Spot-fix vs systematic-fix; complexity has a cost.

AI makes spot-fixes so cheap that you keep reaching for them when systematic answers would be better. Write-cost dropping doesn't make maintenance-cost drop with it.

Order of adoption

1. **CD + observability/SLOs as paired foundation.**
   Each without the other is half a story.
2. **Code ownership + CLAUDE.md and skills.**
   Without these, AI amplifies confusion.
3. **Hermetic, speedy builds + feature flags as you scale.**
   Reproducibility for verification; gradual rollout for safety.
4. **Chaos engineering as audit, not foundation.**
   Once the rest is solid.



Structuring the platform team

- **Classic platform work + maintaining the agent substrate.** CLAUDE.md, MCP servers, AI-legible CIs, dissemination tooling, auto-review rules.
- **Intercom's pattern: "team-2x" + plugin marketplace** (hundreds of contributors & skills). Modernising the factory is everyone's job.
- **The platform team maintains the agent's relationship to the platform.** Not just the platform.

Three things to take away

AI amplifies your existing practices. Going fast without autonomy, ownership, and feedback loops is enshittification.

Be deliberate about capacity allocation. Reinvest the new throughput in the platform substrate that lets you absorb more, not in marginal features.

"We didn't obviously break anything" is not success on its own. No one in this room is done. Including us.

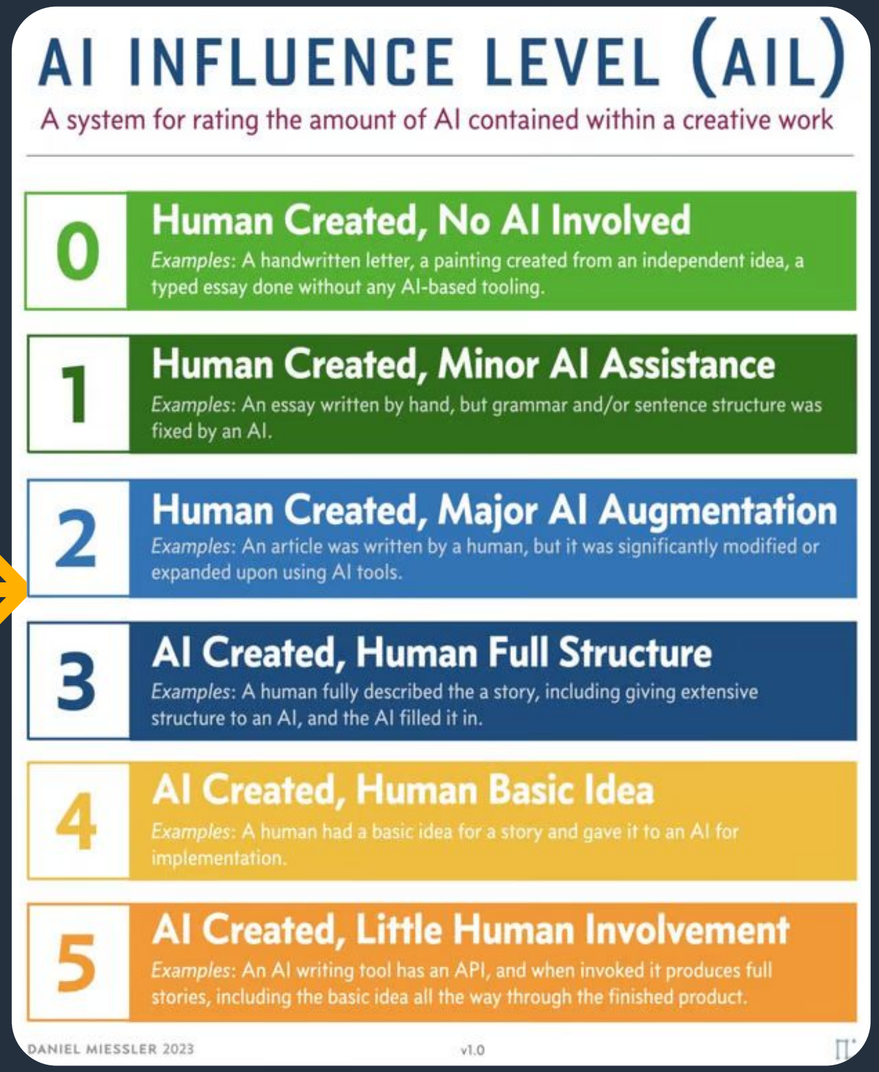
Thank you.

danielmiessler.com/blog/ai-influence-level-ail

Slide text & speaker notes: AIL-2.5

Images: AIL-0 (by bbghost.bsky.social)

Art should be made by artists, not machines.



honeycomb.io

Liz Fong-Jones

Technical Fellow | Honeycomb

Sydney, NSW | MACS (Snr) CP

linkedin.com/in/efong

lizf@honeycomb.io

@lizthegrey.com