What Staff Engineers Do Differently

Patterns from 12 years of calibration rooms at Meta - why some engineers get promoted while others stay stuck, no matter how hard they work.

Konrad Michels
Key Career Coaching
12+ years at Facebook/Meta. Hiring committee member. Jedi, TPM, and Infrastructure System Design interviewer. Led Project Sandstorm - Facebook's first planned datacenter shutdown with zero user impact.

1. The Behaviors That Separate Staff from Senior

They solve problems nobody assigned them

Senior engineers execute assigned work well. Staff engineers notice unassigned problems and act without waiting for a ticket or permission. At Meta, a senior who delivered everything perfectly could lose in calibration to a staff engineer who delivered slightly less but also quietly fixed a deployment pipeline blocking three other teams, wrote the RFC resolving a year-long architectural disagreement, and caught a database migration before it caused an outage.

Their unit of concern is the system, not the feature

When I was building the EU SRE presence at Facebook, the engineers who grew into staff roles weren't focused on getting their own services to pager-acceptable reliability. They were asking about cross-region consistency, about which US playbooks were solving EU-specific problems vs. just copied because familiar. They were thinking about the system before the system existed yet.

They write things that outlast them

Design docs, postmortems that explain contributing factors rather than just listing action items, internal guides that become the reference three years later. At Meta, engineers whose impact was clearest at promotion time almost always had a document trail. Managers could point to artifacts: docs that shaped decisions, postmortems that changed processes, design proposals other teams adopted.

They say no to work that doesn't matter

I watched senior engineers turn down the staff promotion they'd been chasing for years because they couldn't stop attending every meeting, picking up every small bug, helping every engineer who asked. They wanted staff-level impact but couldn't let go of senior-level habits. Saying no is a skill, not a personality trait.

They make other engineers better

Not "good mentor" in the vague sense. The engineer whose code review comment makes a junior engineer think differently about a whole class of problems. The engineer whose design docs become the team template because they're so clear. In calibration at Meta, this showed up as "when this engineer is in the room, the room produces better outcomes."

The common thread: All of these require acting on your own judgment before you have external validation. Senior engineers wait to be recognized at staff level before behaving like staff engineers. Staff engineers start behaving like staff engineers and get recognized as a result.

2. How Promotion Actually Works (Inside the Calibration Room)

At the end of a review cycle at Facebook, your manager walks into a calibration meeting with a stack of cases. Every other manager in the room either supports the case or pokes holes in it. The managers poking holes haven't worked with the candidate. Skepticism is the default.

A manager who can say "she led the migration that unblocked three other teams, and I have two other managers here to back that up" is in a fundamentally different position than one saying "trust me, they did great work."

The 5 people who decide your promotion

Your skip-level. In the room alongside your manager, expected to have an independent view. If they don't have an independent impression of you before calibration starts, your manager is carrying the whole argument alone.

Peer Staff and Principal engineers. There specifically to validate whether your work genuinely operates at the level being proposed. They're looking for a pattern of judgment: defensible decisions under ambiguity, pushing back when you had good reason and being right.

Cross-functional leads you've worked with. Not usually in the room, but their voices get there through your self-review and your manager's case. A PM saying "we wouldn't have shipped without the architecture decisions [your name] made" is different evidence than your manager vouching alone.

The skeptic. Every calibration has one. They ask: "What's the system-level impact?" "Could a strong Senior have done this with the right support?" Promotions get deferred most often not because the work was bad but because the case wasn't built to survive scrutiny.

You, through your artifacts. Not in the room, but present through every piece of evidence that exists before anyone says your name. Written communication, design docs, postmortems. If the artifact stack is thin, the summary is thin.

> **Key insight:** Promotions are resource allocation decisions made by people who haven't seen your work, based on a packet skimmed in 30 seconds, advocated by a manager competing for limited slots against four other managers with similar cases.

3. The Visibility Problem

Engineers who are technically brilliant and genuinely excellent at their jobs stay stuck at Senior because no one above them can see the work. They do the work. They just do it quietly.

Why "invisible work" happens

Depth without breadth. You're the expert in your service, know it better than anyone. But that expertise is visible only to people who already work with you. Outside your team, you're just a name in a code review.

Reactive work. You solve problems as they come. The impact is real but there's no artifact, no document, no decision anyone can point to later.

Heroics without documentation. You pulled the team through a rough quarter, stayed late, filled gaps. Six months later when someone tries to reconstruct what happened, there's no record of your specific role. The incident says "the team."

What "increase your scope" actually means

Your work needs to create visible artifacts that cross team boundaries. Other people, outside your team, need to be able to point to something you did and say: that helped us. A design doc that multiple teams adopt. A framework two other teams now use. An architectural recommendation that changed how an adjacent team approached a problem.

Visibility isn't self-promotion. Self-promotion is telling people how great you are. Documentation is creating a record of what happened. There's a difference, and calibration committees can tell.

Practical moves

- Write a one-pager before you do the work, not after. Circulate it to anyone affected.
- Pick one cross-team problem per half and own it explicitly.
- Ask "what was the specific pushback on my case in the last calibration?" not "what would it take to get promoted?"
- Stop fixing things silently. Write down what happened and why it was non-obvious.

4. The Artifact Mindset

If there's no artifact, it didn't happen. The people making promotion decisions can only evaluate evidence they can see. Your manager has eight other direct reports. Your skip-level has forty engineers. The promotion committee has never worked with you directly. They will evaluate you entirely on what's written down.

Strongest vs. weakest promo packets

I reviewed promotion packets at Facebook for years. The strongest had a clear thread: "here's the problem, here's what I built or decided, here's what changed." Created throughout the year, not reconstructed from memory.

The weakest were reconstructions - an engineer trying to reverse-engineer impact from deploys and calendar entries. Approximate numbers, assembled rather than observed. Promotion committees can tell the difference.

The engineer whose promo didn't go through (then did)

A strong engineer at Facebook - "the kind of person who always knew what was happening across the systems they owned" - was asked "what did you do this year?" at promo time. Answers existed. Almost nothing was documented. The promo didn't go through.

The following cycle, same engineer got promoted. Not because they did more impressive work. Because they spent six months building documentation habits. The artifacts were specific, timestamped, clear. The work hadn't changed. The visibility had.

What counts as an artifact

- A design doc written before starting a project - even a short one. Has a timestamp. Shows your thinking.
- A post-incident review where you led the analysis. Your name is on it.
- A short write-up sent to your team after a migration: what we did, what changed, what we learned.
- An email flagging a risk six weeks before the incident that didn't happen because you caught it.
- A data pull shared with your manager showing what the improvement actually looked like.

Build the habit

- Write a project note when things close (15 minutes, what happened, what changed)
- Make decisions legible in the moment (a Slack message, a doc comment, a one-paragraph note)
- Share results when they land - "the migration is complete, we're seeing X improvement"
- Keep a weekly log - five minutes, two or three things done. Over six months it's a detailed career record.

5. The Mistakes That Keep You Stuck

Working harder instead of differently. Engineers who ground 60-hour weeks, covered every on-call, never said no - almost never made Staff. They were too valuable exactly where they were. Meanwhile someone doing what looked like less work kept moving up because they were working on problems the skip-level cared about.

Staying within team boundaries. If your work has been contained within your immediate team, your manager walks into calibration alone, vouching for you against people who have no reason to believe it.

Asking the wrong question. "What would it take to get promoted?" gets a vague answer. Better: "In the last calibration cycle, what was the specific pushback on my case?" If your manager doesn't have an answer, that's also data - your name might not be coming up at all.

Holding onto senior-level habits. Saying yes to everything, attending every meeting, picking up every small bug. These habits that built trust at the senior level become the exact behaviors that prevent Staff-level impact.

Relying on reactive work. Solving problems as they come is valuable but creates no visible pattern. The non-incidents - crises that didn't happen because you caught something early - are the most invisible work of all.

> **The fundamental shift:** A Senior engineer makes their team better. A Staff engineer makes other teams better. When a Staff engineer has a great quarter, adjacent teams know it, and those teams' managers can speak to it.

Want help applying this to your situation?

This guide covers the patterns. Your career has specifics - a particular company's promotion process, a manager relationship that may or may not be working, a technical portfolio that may need repositioning.

Book a 30-minute call - $25

Honest feedback on where you stand, what's working, and what to change. No sales pitch, no fluff - just a direct conversation with someone who's been on the other side of the table.

keycoaching.co

Self-guided systems

The same frameworks I use with coaching clients, packaged as step-by-step guides you can work through at your own pace.

resources.keycoaching.co