GRAHAM GILBERT / PSU MAC ADMINS 2025

THE PATH TO STAFF ENGINEER AND BEYOND

STAYING ON THE IC TRAIN

The views expressed in this talk are my own and do not necessarily reflect those of my employer

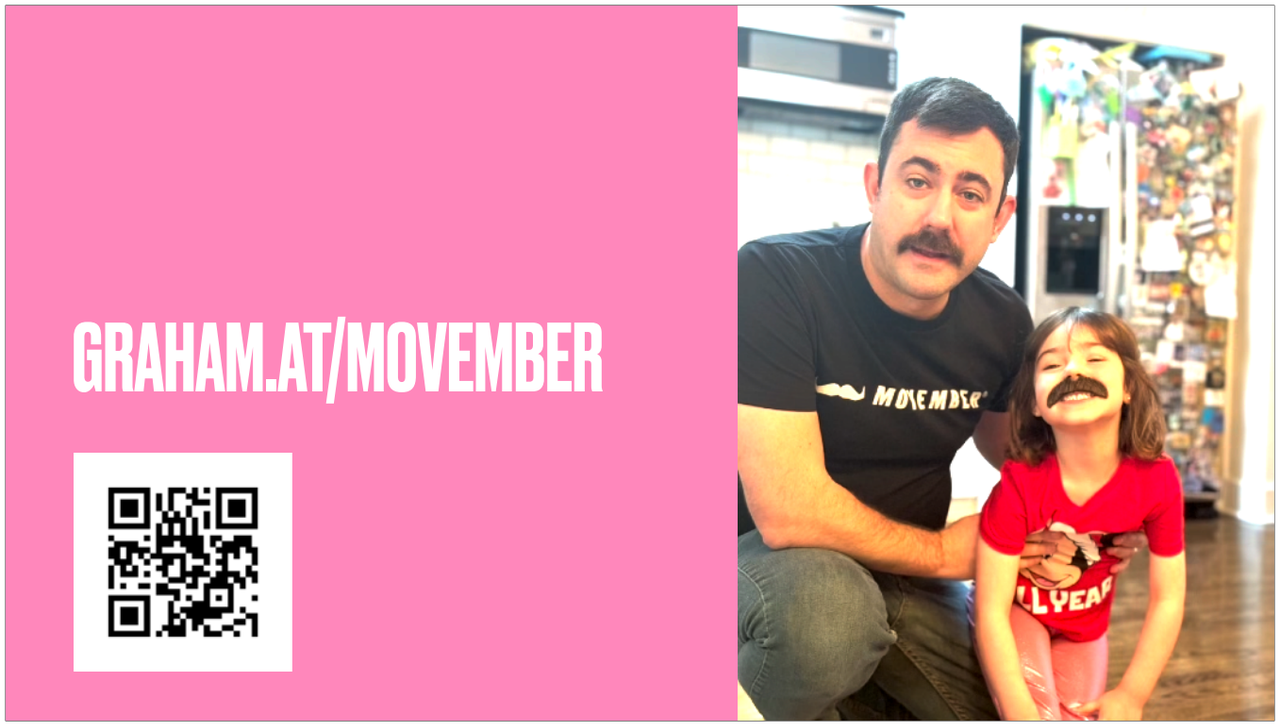
Before we start, a quick disclaimer: The views expressed in this talk are my own and do not necessarily reflect those of my employer

HELLO!

- I’m Graham
- Senior Tech Lead Manager at Airbnb
- Recovering open-sourcer

Hello there - my name is Graham Gilbert. My day job being a senior tech lead manager on the Client Engineering team at airbnb - a senior TLM is either the best or worst, depending on your point of view, of half senior manager and half senior staff engineer. I have written several open source tools over the years, including Crypt, a FileVault escrow tool, the macadmins osquery extension and the now defunct Imagr.

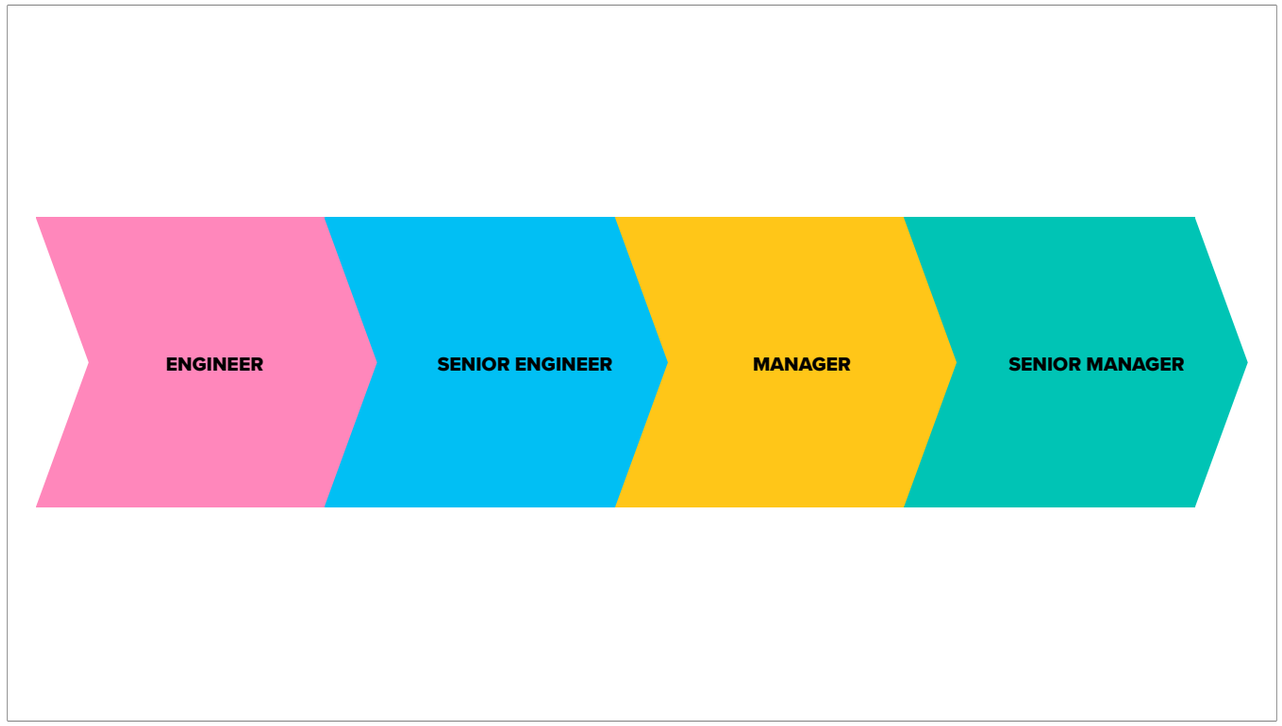
I am on the board for Mac admins open source, and we do things like supporting the SOFA feed and signing munki. Running Mac admins open source isn’t free. We have a not inconsiderable cost for the hosting and bandwidth for Sofa in particular. If you use Sofa, and find it useful, or find having a codesigned Munki or osquery extension useful, or even use any of our osquery tables in your commercial product, please consider making a small donation to help cover our costs. The board have been covering a lot of the costs ourselves because we really believe in what we are doing, so the community stepping up and helping us out would be lovely.



Finally, I am a testicular cancer survivor and Movember fundraiser. So far, we have raised over $50,000 that goes to funding research and awareness campaigns for a whole host of mens health issues, including testicular cancer and prostrate cancer. Fun fact: My daughter's birthday is in November, and this past year she asked that instead of her friends bringing presents to her birthday party, she wanted them to make donations to Movember. So a six year old is making the right call in helping put a stop to men dying too young, so perhaps if you find this talk useful, you could also consider donating.

So let’s get into it. You’re a Mac Admin. You’ve probably been doing this for a while. You’ve built systems, automated the boring stuff, solved hard problems, maybe even mentored junior folks. You know your way around this stuff. But now you’re asking: What’s next?

And if you’ve looked at the org chart, you might’ve noticed that it looks like a glass ceiling is looming over you. You go from engineer… to senior… and then? Manager?



At many organizations, the career path from engineer ends up in management. If that is a goal of yours, then that is great. But management isn’t for everyone. The sad fact is that in a lot of places, people are forced to move into management if they want to progress. And they’re forced to move into management without being told the big secret about being a manager

BEING A MANAGER IS A COMPLETELY DIFFERENT JOB

Managing people is a completely different job to being an engineer. Some engineers are good people managers. Some engineers are terrible managers - the same as some managers are terrible engineers. So, if being a manager isn’t the route you want to take, what else is there?

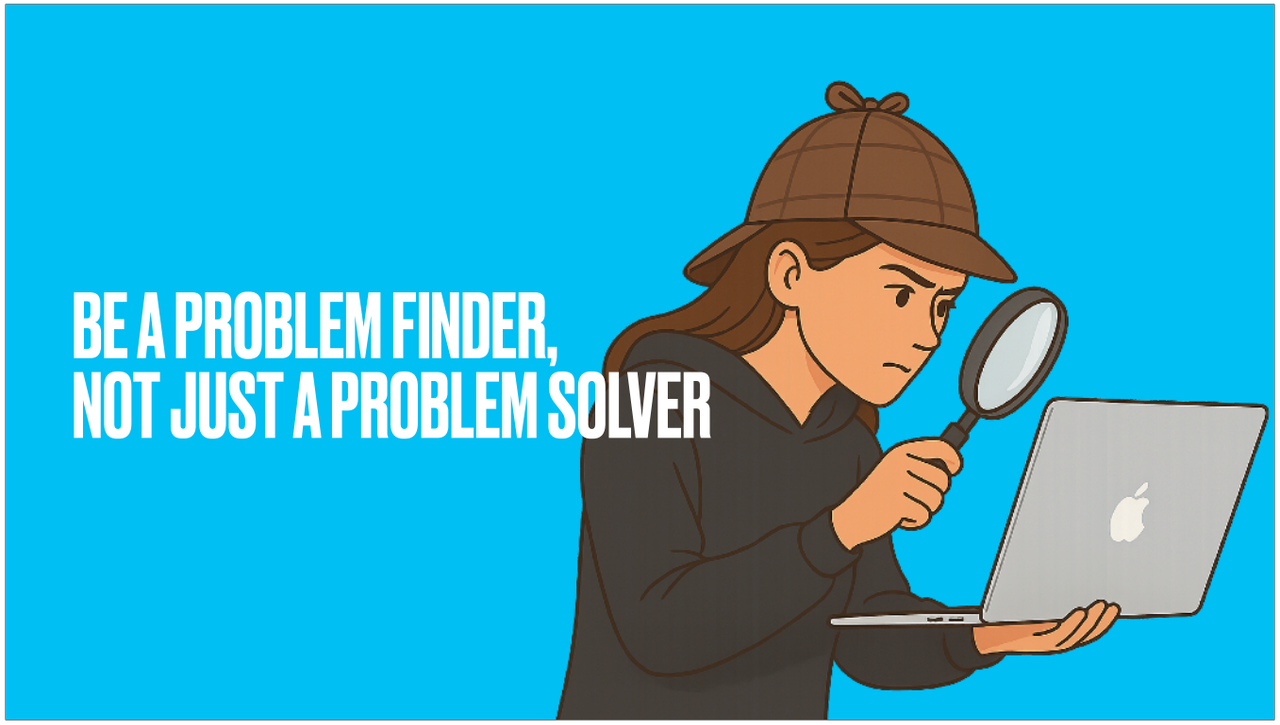
STAFF ENGINEER

For the observant amongst you, you won’t be shocked to learn that the next step up the leadership ladder is a staff engineer. But what is a staff engineer? What makes them different from senior engineers?

The big shift is in scope and impact. A Staff Engineer is thinking about the long game. Not just the next sprint or the next project — but the next year, or even the next three years. You're setting technical direction. You're shaping systems, not just building features. You're identifying cross-team risks, defining architecture patterns, and driving alignment at a higher level.

You're expected to think bigger, wider, and longer-term than the rest of the team — and help others execute toward that vision.

So what else differentiates a staff engineer from others?



Let’s talk about a mindset shift that’s absolutely critical if you want to move into a Staff role. As engineers, we’re trained to solve problems. Something breaks — we fix it. There’s friction — we automate it. And that’s great. That’s valuable. But at the Staff level, that’s the baseline. The real shift is learning to become a problem finder.

That means looking beyond your immediate tasks or team. It means looking for patterns, gaps, inefficiencies — things that might not be broken yet, but will be. Things nobody else has noticed, or has the perspective to frame as a problem. And then, critically: deciding which ones are worth solving. Not every weird problem or outdated workflow needs a fix. But part of your job is knowing which ones do, and having the influence to do something about it.

So don’t just wait for work to be handed to you. Get curious. Get proactive. Ask: what’s slowing people down? Where are the gaps between teams? What’s fragile, or scaling badly, or a security risk no one’s flagged yet? That’s how you stop reacting and start leading.

Another hallmark of a great Staff Engineer: they’re a force multiplier. Your impact shouldn’t just come from the code you personally write. In fact, the more senior you get, the less of your impact should be directly tied to commits. Your job becomes making other people more effective.

That could mean unblocking a teammate who’s stuck. It could mean writing clear technical docs that help others make good decisions without you in the room. It might be mentoring someone so they level up faster — or influencing architecture so your whole org avoids a bad decision. You might lead by building shared tooling, or even just creating a clear mental model that helps your team reason about a complex system.

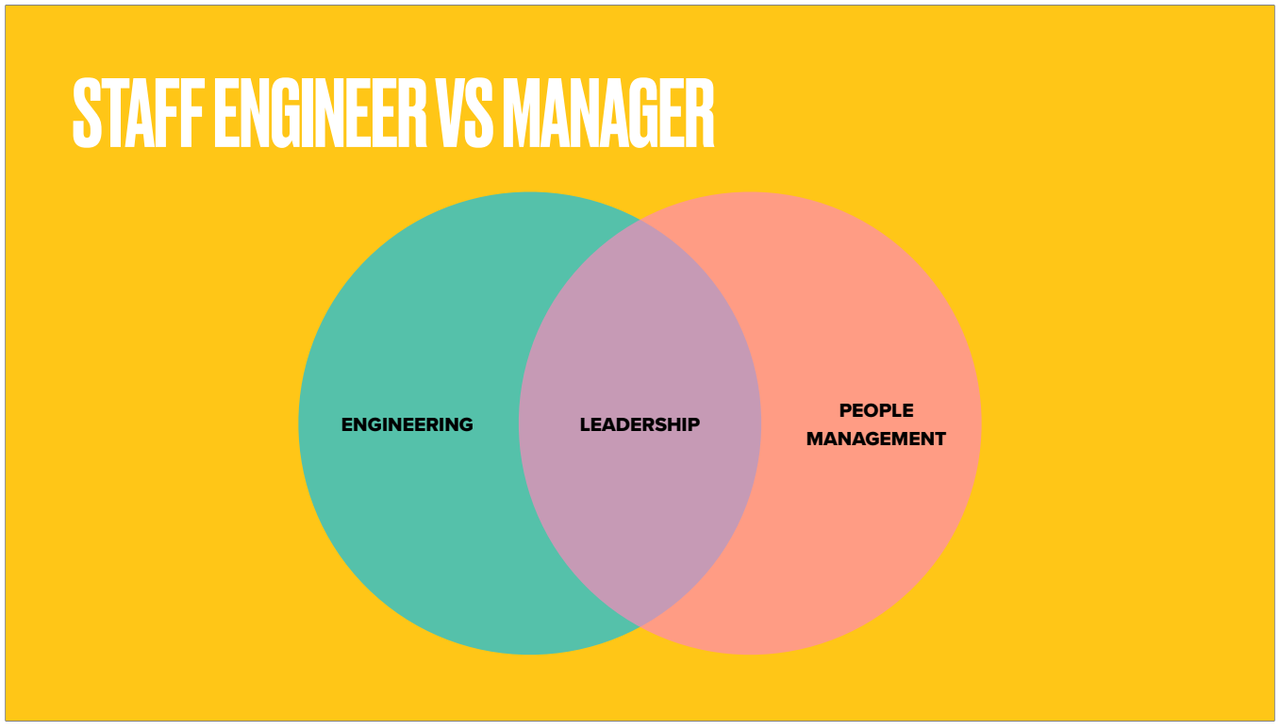
The best Staff Engineers leave behind a wake of better engineers, smoother systems, and smarter decisions.

WAIT - THAT SOUNDS LIKE BEING A MANAGER

Now, you might be hearing all this — coaching people, unblocking teammates, driving alignment — and thinking:

“Wait a minute… that sounds a lot like being a manager.”

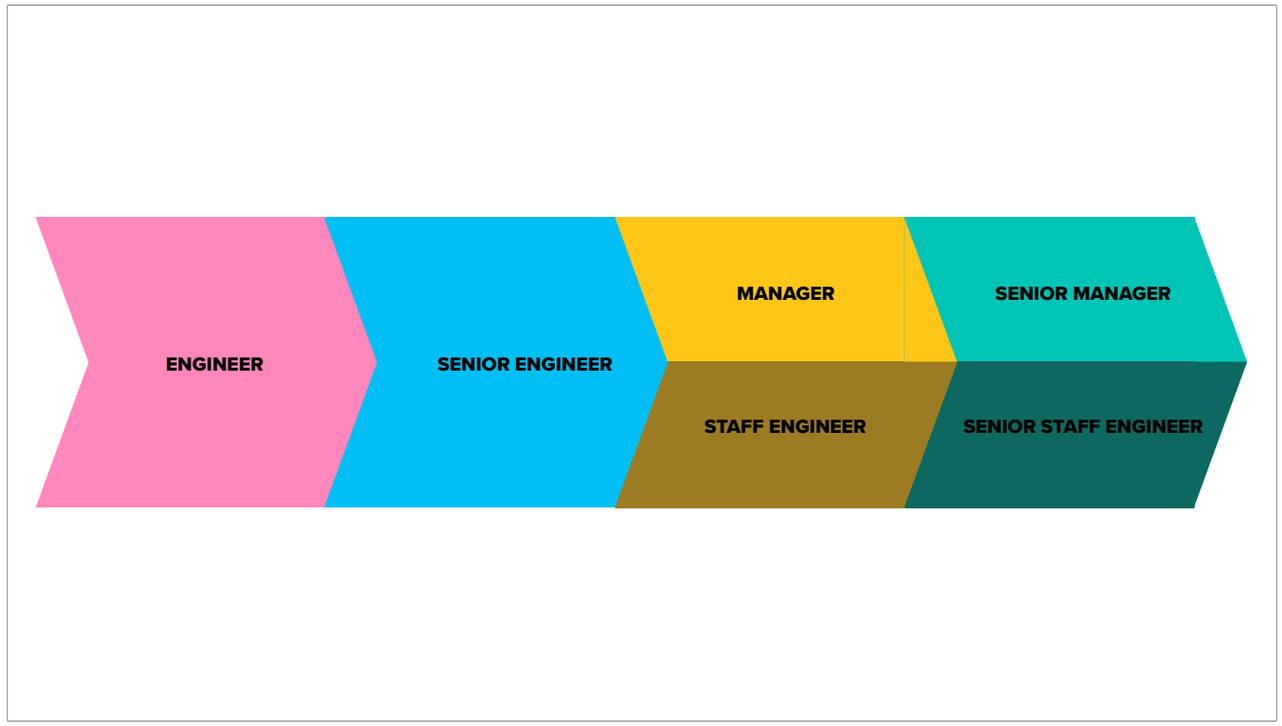
And… yeah. You’re not wrong.



There is overlap. Both roles require leadership, influence, communication, and vision. But the how and what are different.

Managers focus on people: hiring, performance reviews, career growth, team health. Their toolset is organizational.

Staff Engineers focus on systems, architecture, and long-term technical direction. Their toolset is technical. Both are leadership roles. But Staff Engineers lead without authority. You’re not someone’s boss — but you still need to earn trust, drive consensus, and move the org forward.



This is how mature organizations think about the two tracks.

Management and technical leadership aren’t a hierarchy — they’re parallel paths. Staff Engineer isn’t a stepping stone to management. It’s its own destination.

Now, not every org gets this right. Some still see management as “the real path to leadership.” But that’s changing. And it’s up to us — as a community — to keep pushing for that recognition.

You can be a leader without managing people. Staff Engineer is proof of that.

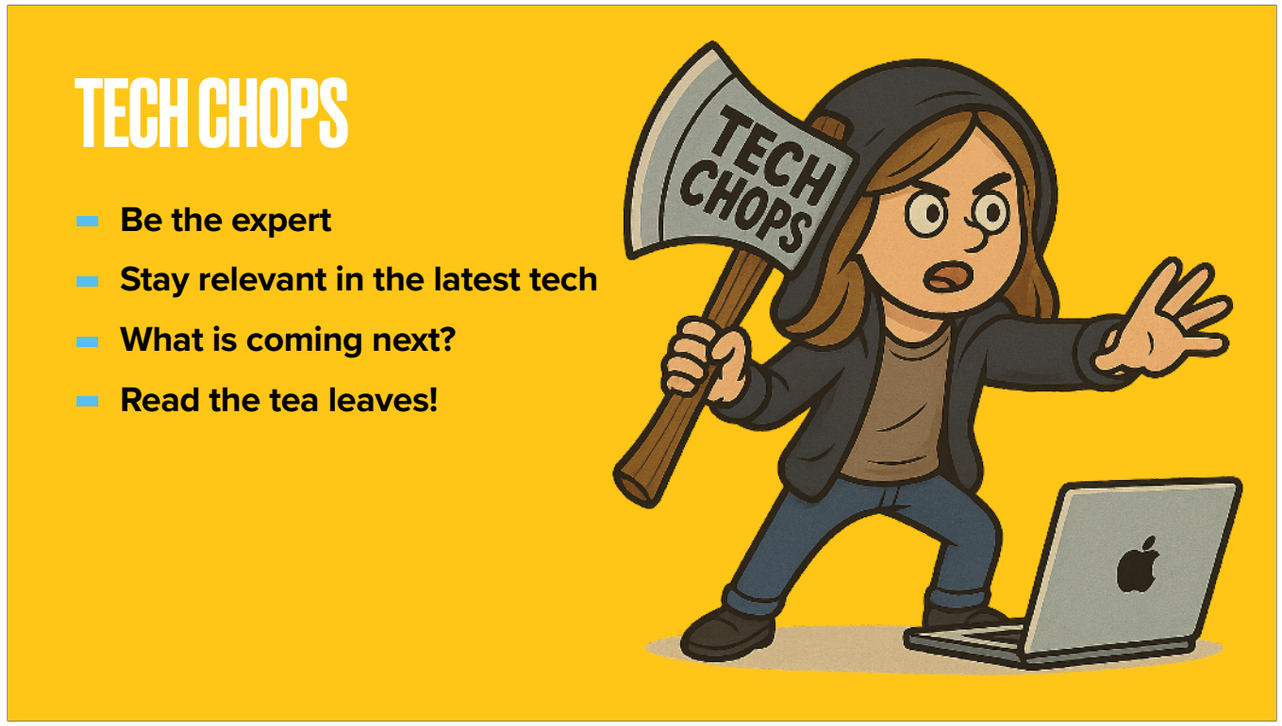
OKAY, WHAT DO I NEED TO BE A STAFF ENGINEER?

Alright — so maybe by now you're thinking: This sounds pretty interesting. I want to operate at that level. But how do I actually get there?

Let's talk about what separates a senior engineer from a Staff Engineer in practice. What are the attributes that actually matter?

I'm going to walk through a few — and yes, some of them will sound soft, or even a little "manager-y." But trust me — every one of these has a huge impact on your ability to lead without authority and drive meaningful outcomes.

Obviously you need to know your tech. As a staff engineer you’re the tech boss. But what does that mean really?



At the Staff level, you're expected to be an expert. Maybe not in everything, but definitely in your area. You should be the go-to person for your domain — whether that's macOS internals, MDM protocols, software deployment, endpoint security, whatever it is.

But expertise today isn't enough. You need to stay current. That means keeping up with the latest platform changes. It means banging on the beta of macOS the moment it drops. Watching WWDC like a hawk. Staying on top of changes in identity, networking, security. Reading between the lines of Apple's sometimes vague documentation.

We work in an ecosystem where we don't get roadmaps. So part of your job is to read the tea leaves. A quick show of hands: Who ordered an iMac Pro even though you had no intention of deploying them in your environment. I did, because it was the first device with a T2 and we needed to see how it would impact us. What about ordering a an Apple Silicon Developer Transition kit to test your tools? Doing this helped reduce the risk of these transitions to our org by getting ahead of these new technologies before they were forced upon us.

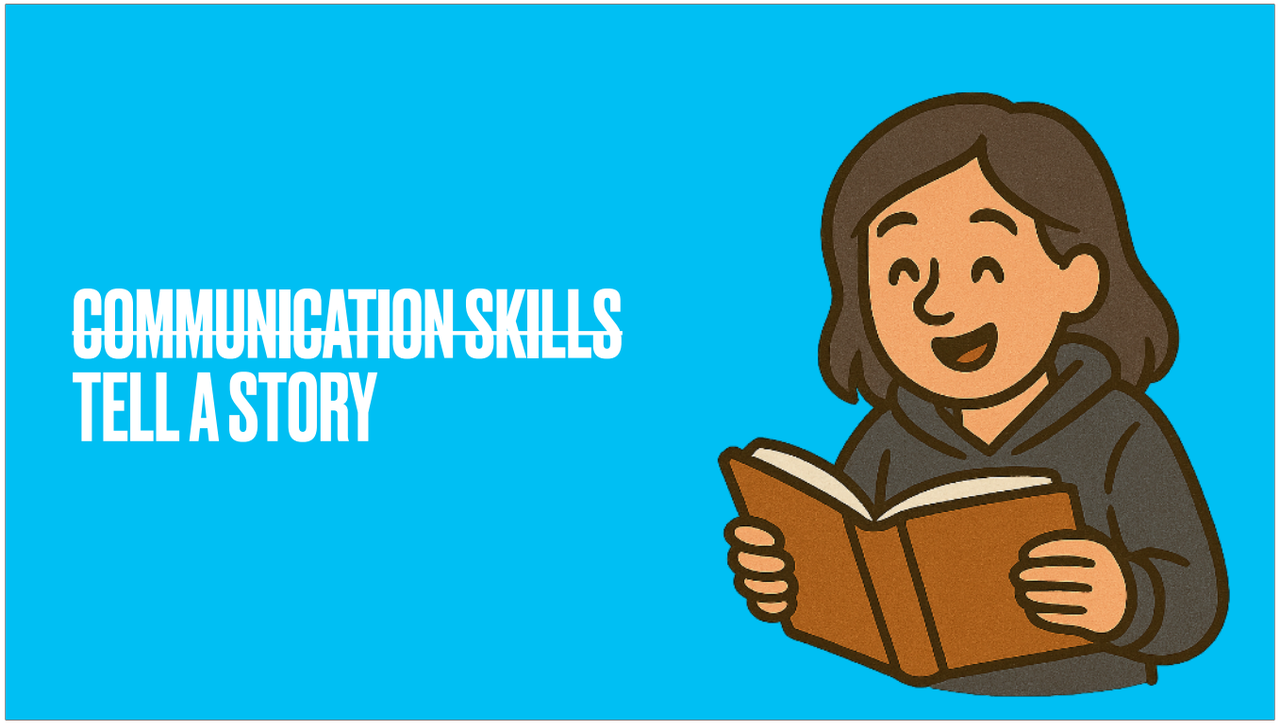
Being technically strong is what gives you the platform to influence — because people trust engineers who know what they're talking about.

COMMUNICATION SKILLS

Okay — so you've got the technical depth. Now what?

Now you need to communicate it.

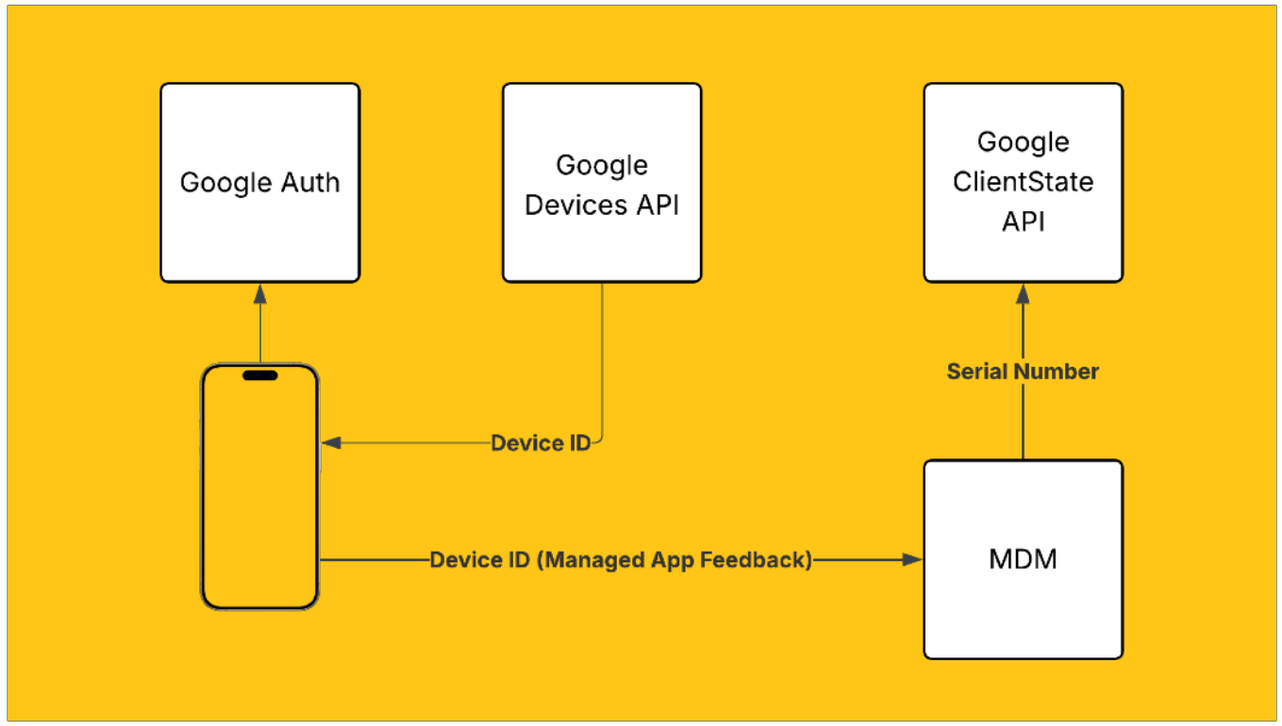
And I don't just mean writing clear documentation or speaking up in meetings — though both are important.



I mean learning to tell a story. Staff Engineers spend a lot of time explaining things — to engineers, to managers, to execs. And you need to tailor the story every time.

If you're working with a junior engineer, you're breaking down complex systems in a way that makes them approachable and actionable. If you're pitching a project to a VP, you're not talking about APIs or MDM or build systems. You're talking about business value. Time saved. Risk reduced. Strategic alignment.

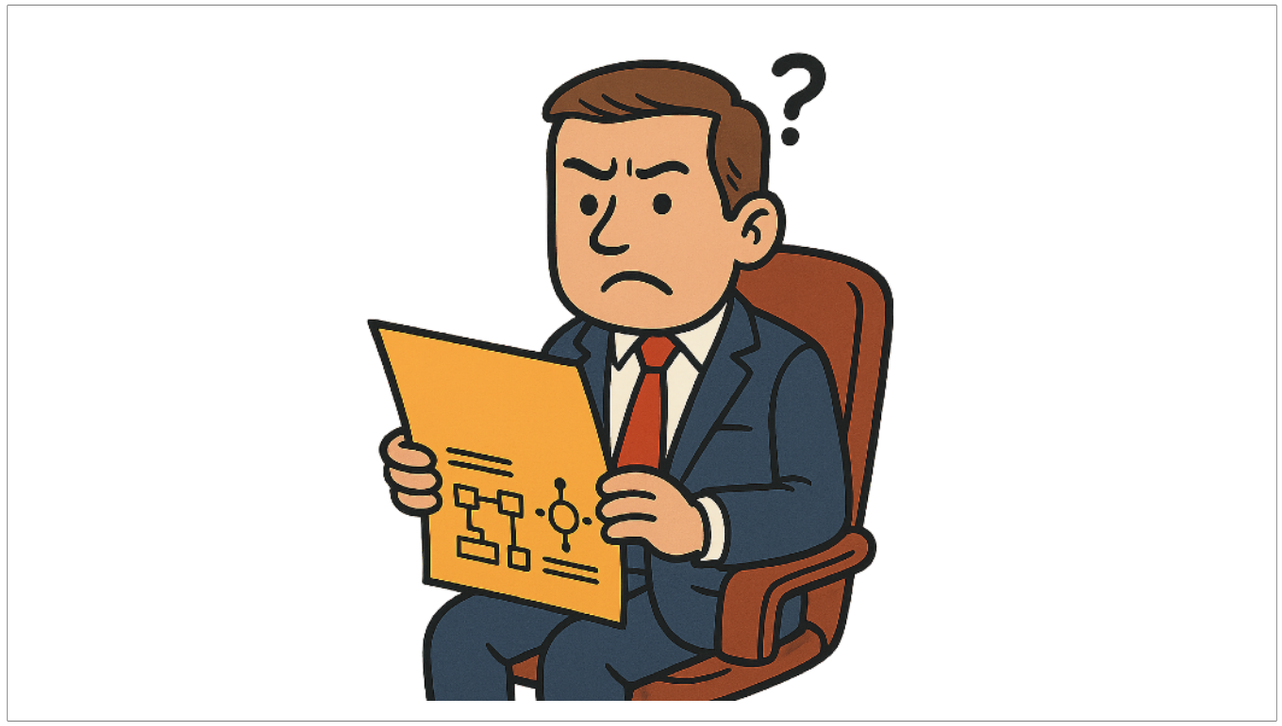
Same problem, different message.



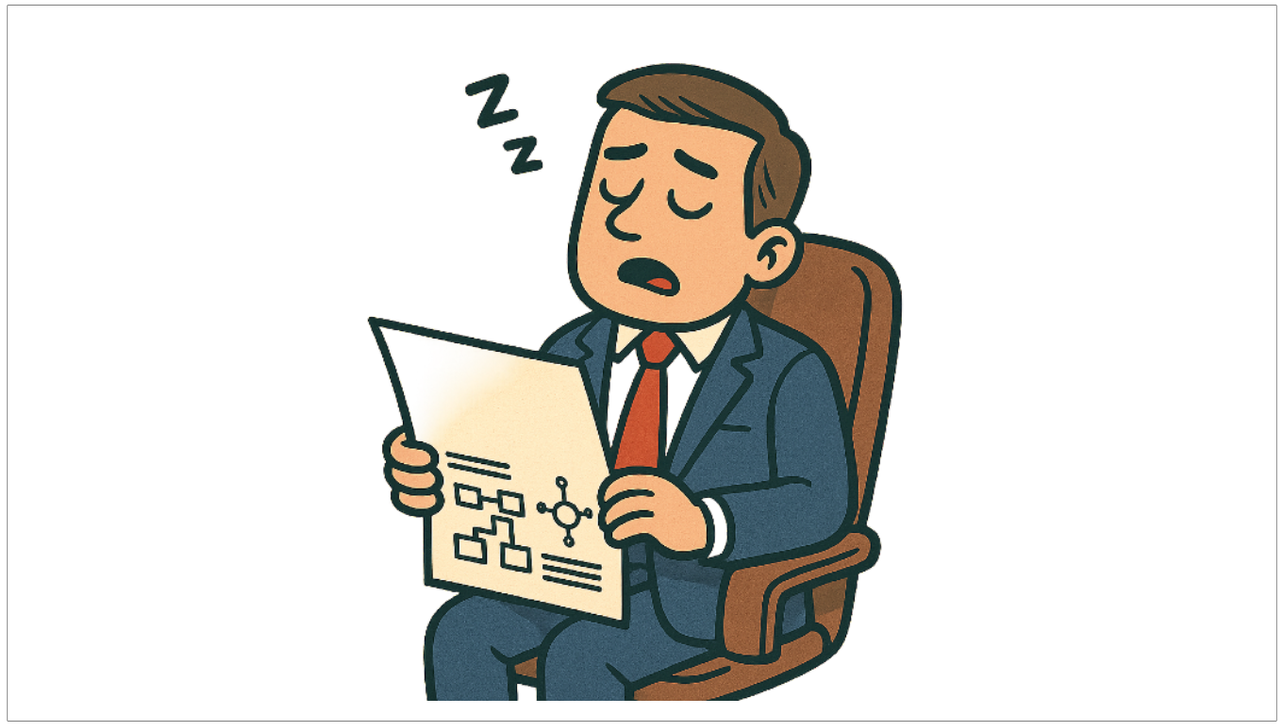
Let’s say you’re proposing to build an iOS app to communicate with end users and integrate with Google context-aware access. You’ve got the design doc, the architecture, the security model.

Great — that’ll make sense to your engineers.

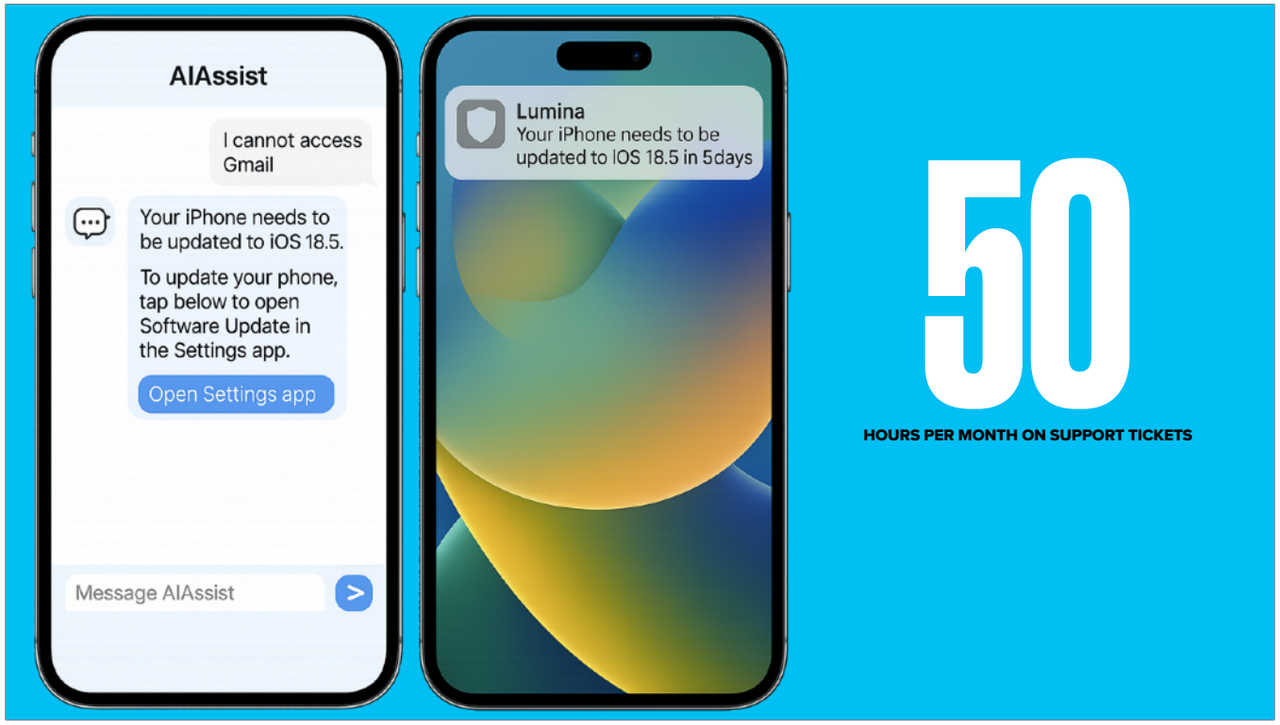
But if you send that same doc to an exec?



Chances are they’re either going to be confused by all the APIs, APNS’s and ClientStates



Or even worse completely bored and uninterested!



You need to speak their language. They want to know what the impact will be - what the improvement to user experience will be, what this will do to our support ticket numbers. Impact is what gets attention here.

Now, let’s talk about opportunity. When you’re early in your career, you’re mostly handed projects. Someone tells you what needs doing, and you do it.

At the senior level, you’re probably starting to suggest projects — maybe you see inefficiencies and you write a script or build a new workflow. But at Staff? Sometimes the work doesn’t look glamorous at first.

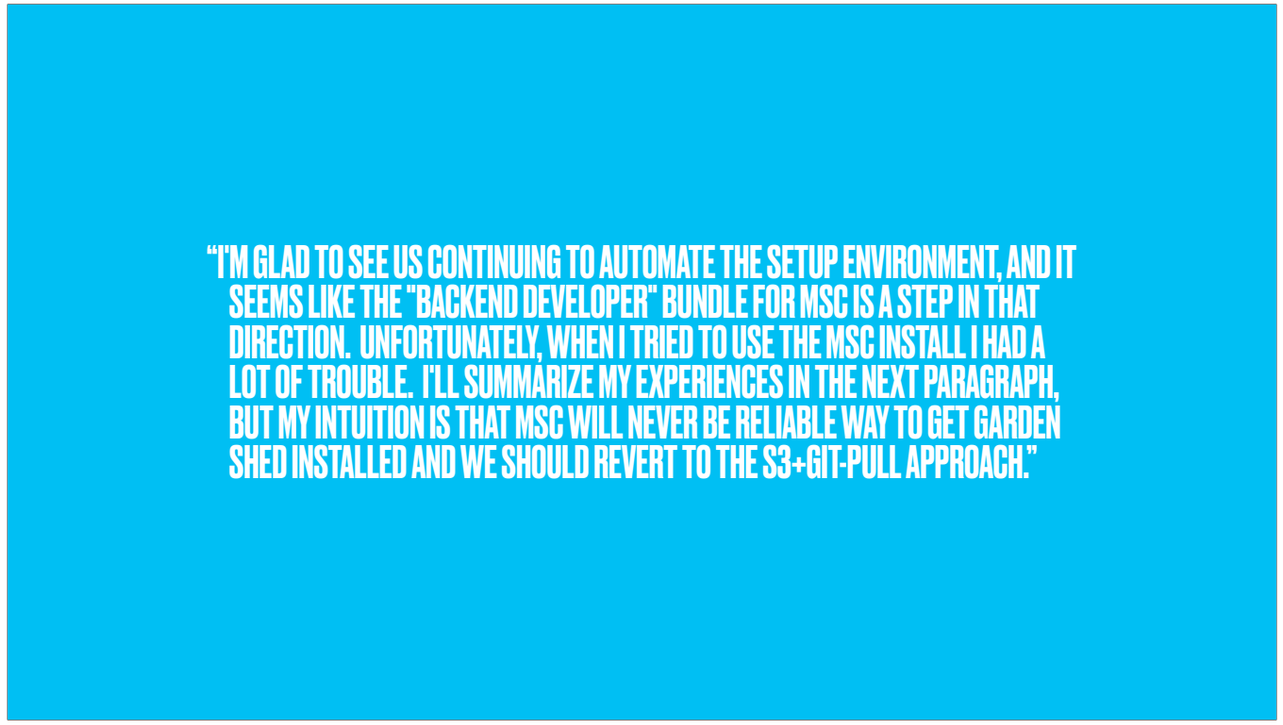
Show of hands: Who here has ever been handed a project that felt messy, political, or way under-scoped?

Perfect. Let’s walk through what that looks like — and how you can turn that into a moment of growth instead of frustration.

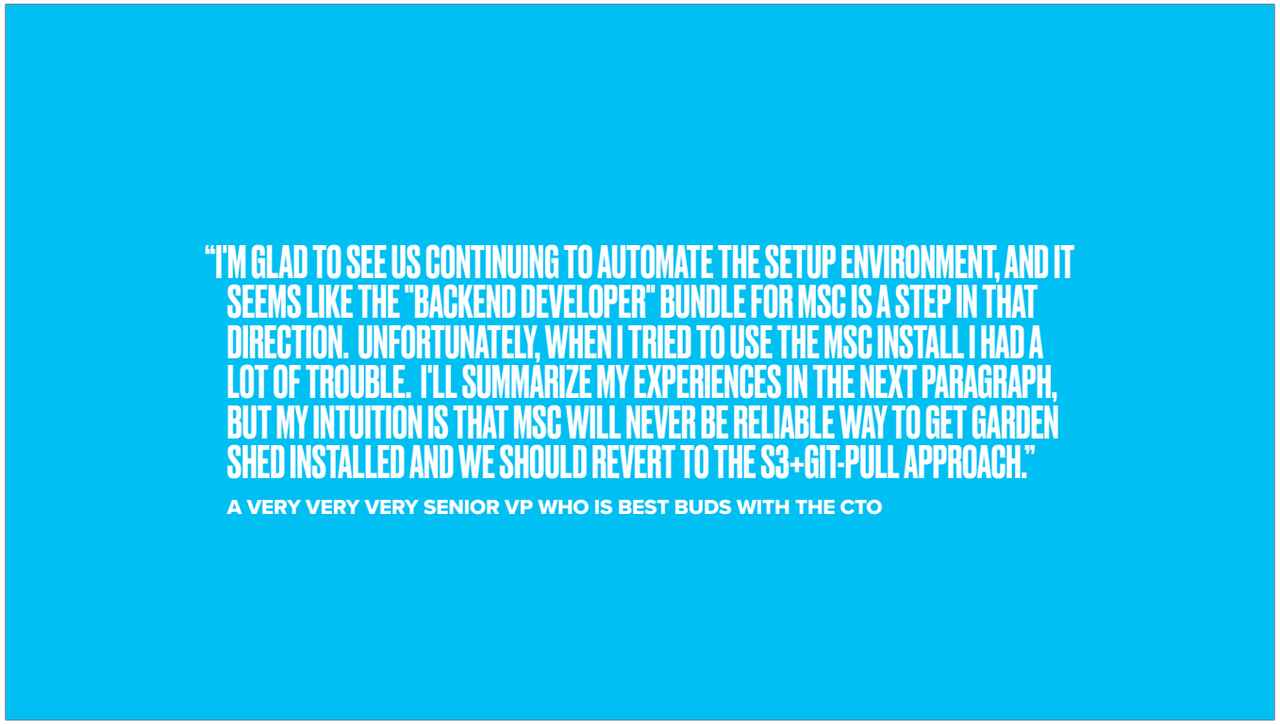
PROJECT: DEVELOPER SETUP AUTOMATION

- Liaise with developer teams for requirements
- Write design documents
- Custom AutoPkg recipes
- Add items to Munki
- Hunt down owners of engineer setup documents
- Get new process added to new engineer bootcamp
- Home for tea and medals

So let’s talk about a hypothetical project. You have been asked to automate the setup of developer laptops. You have spent time talking to the owners of the dev tools to gather requirements, you’ve written design docs, you’ve done a ton of work to get it implemented, and it’s in production. You’ve cut down a 30 or 40 step process down to a handful. You’re feeling pretty good about yourself.



Then this sort of feedback drops in. It’s not just one paragraph as the intro suggests, it’s at least a couple of pages long. You’re like, “okay, that’s annoying. Someone is pooping over all my hard work.” You’re probably feeling a bit upset about it. {DRINK BREAK, LET THEM READ IT}



But then you realize who it came from. It’s someone immensely important. You’re now in full on defense mode. You can’t ignore this. You’ve got to say _something_.

It's important to note that 271 devices have installed this via MSC since it was added in Q4 of 2024. Besides a bug that was reported during the initial rollout and this feedback, we've received no other reports of issues from Support, DevEx, or feedback on the Quickstart docs.

Issues with Documentation

The user noted several issues with the getting started docs in general, mostly related to outdated or duplicated information.

Beyond us writing the initial Quickstart section detailing the steps to install Backend Developer Tools via MSC, DevEx owns these docs, and they're aware of this feedback.

So you start writing down your defense. You say we've not received any negative feedback, and the complaints you had about the docs? Well, they're someone else's problem. {PAUSE} So let's take a step back and think about how this exec is going to take this response. After all, they spent time writing literally two pages about this, so they obviously care about it even if they're frustrated.

It's important to note that 271 devices have installed this via MSC since it was added in Q4 of 2024. Besides a bug that was reported during the initial rollout and this feedback, we've received no other reports of issues from Support, DevEx, or feedback on the Quickstart docs.

DEFENSIVE

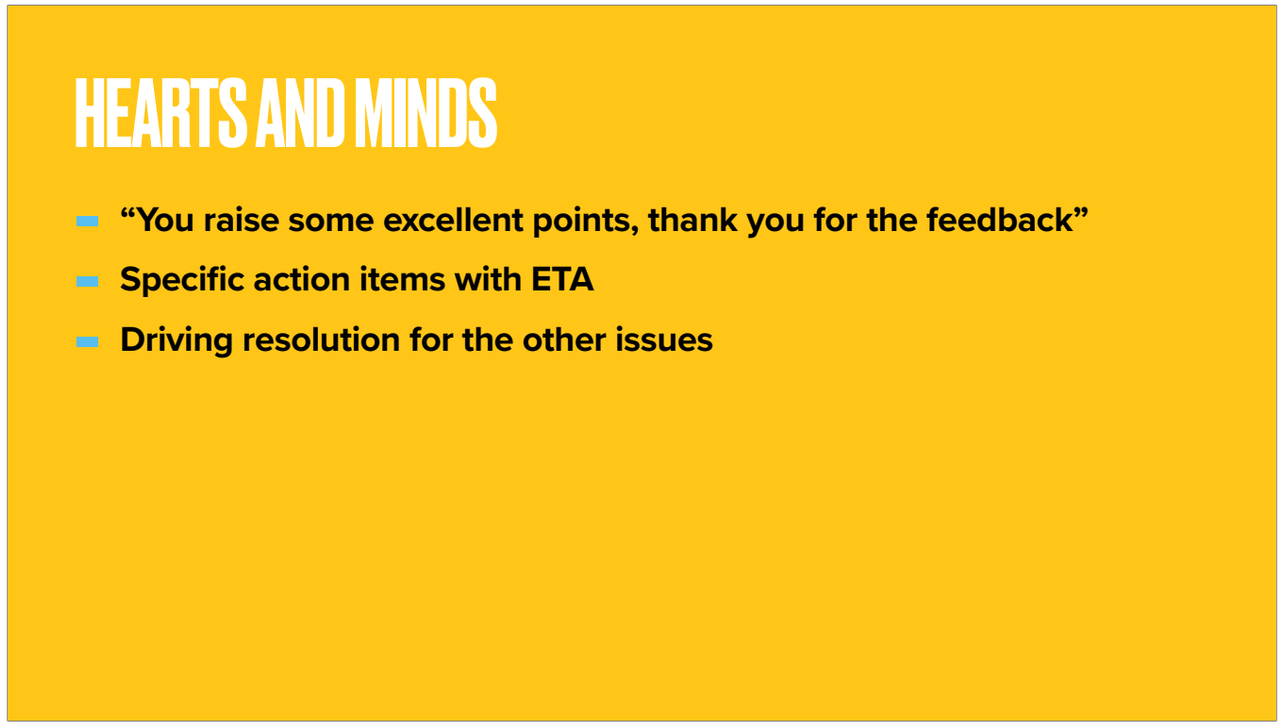
SOME OTHER TEAM'S PROBLEM

Issues with Documentation

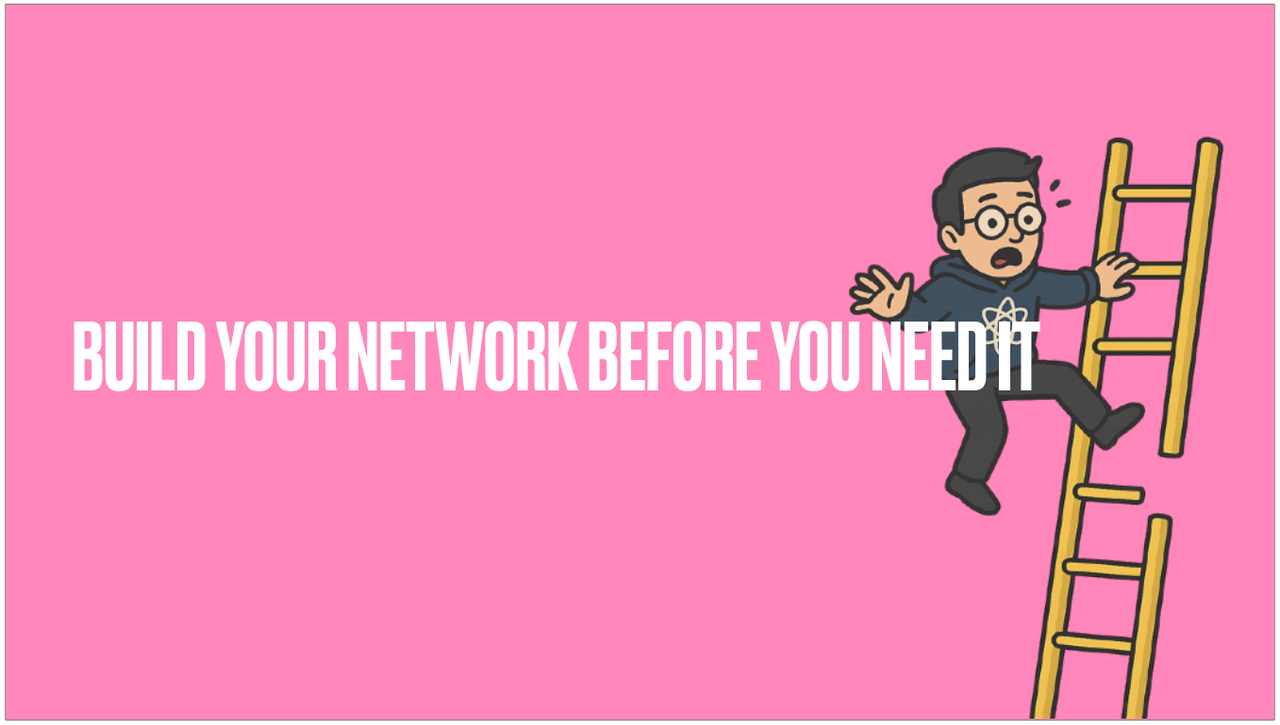
The user noted several issues with the getting started docs in general, mostly related to outdated or duplicated information.

Beyond us writing the initial Quickstart section detailing the steps to install Backend Developer Tools via MSC, DevEx owns these docs, and they're aware of this feedback.

You’re basically saying here we’ve fixed everything we can, the rest isn’t our fault. So at best the exec is going to shrug their shoulders, not have a particularly good opinion of you or your team and move on. At worst, they’re going to continue to insist that your solution is the wrong one, that your team is wrong and doesn’t know what they’re doing. So how can you turn this into an opportunity? How can you cover yourself and your team in glory?



So it turns out that some of the items they raised have already been fixed in a later release. Rather than just saying that, you can acknowledge that they were absolutely right, highlight what you did to fix their issues, and if they have any other unresolved points, be specific about what the next steps are, and **when** you expect to address them. Validating what they say paints you in a much better light, giving specifics about when you’re going to fix it makes you look even better. But the real opportunity here? To corral the other teams responsible for the other 60-70% of the complaints and make **them** fix things. It’s honestly not a ton of work to sit in a meeting or two and explain what they are missing, and to get them to commit to a date to fix it. But what do you get out of it? In this fictional VP’s eyes, **you are the person who got this sorted for him**. You are someone who owns outcomes, not just deliverables. You are now the hero of this story.

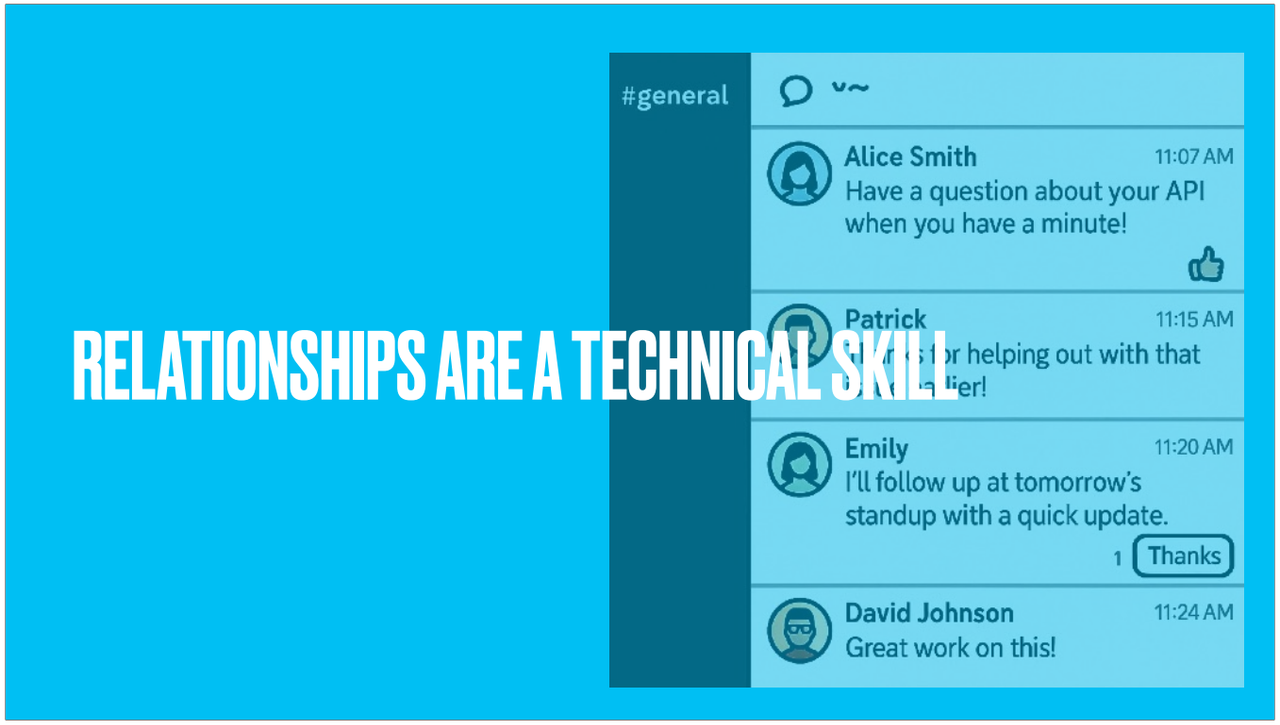


One of the most underrated Staff Engineer skills — and one that I completely ignored early in my career — is building a strong internal network.

Here’s the trap: most of us only start reaching out to other teams after we need something. We’re blocked. We need a change upstream. We want buy-in on a new process. And we’re stuck waiting on people we’ve never talked to before.

Don’t let that be you. Build the network before you need it.

Staff Engineers aren’t lone wolves — they’re deeply connected. Not just to their direct team, but across the org. That’s how they get things unblocked, how they build momentum, how they even know where to spend their energy.

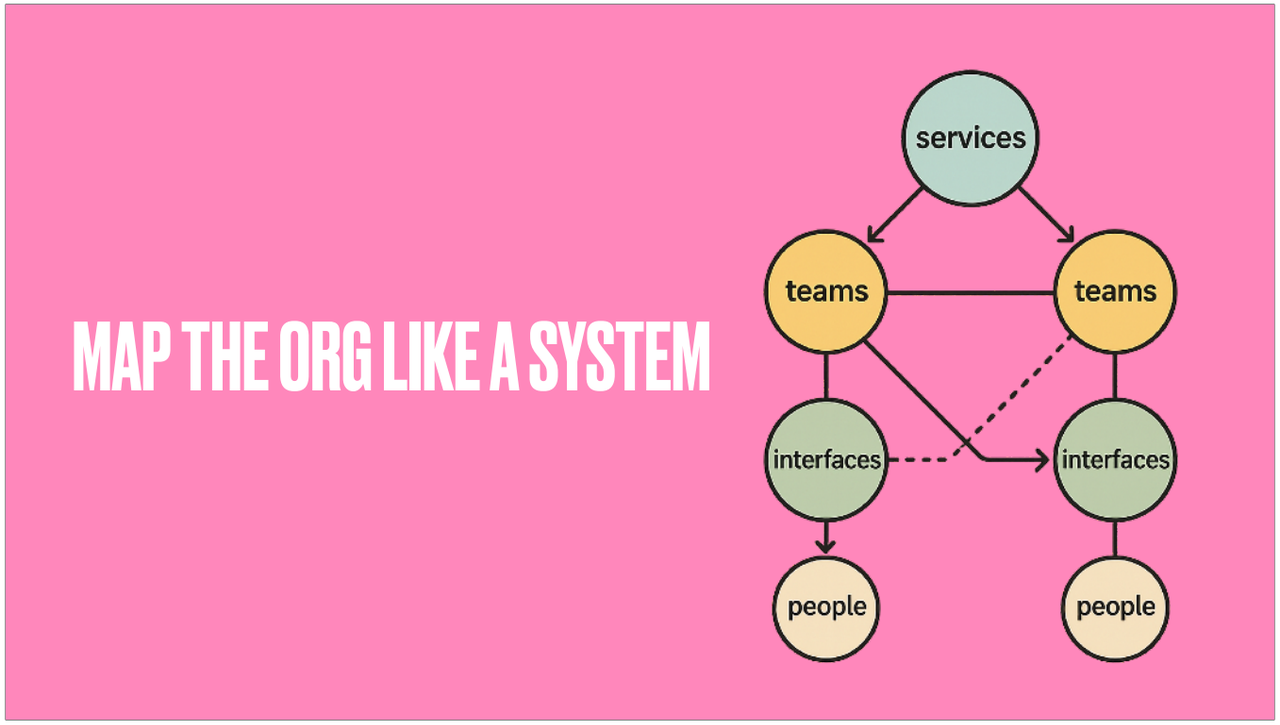


I used to think “networking” was just for PMs and managers — something you did at an offsite or a conference. But here’s what I’ve learned:

Relationships are a technical skill.

Because you can have the best idea in the world, but if you don’t have trust, access, or context — it’s not going anywhere.
And like any skill, you can get better at it.
- Ask questions publicly in Slack so others learn too.
- Follow up with results when someone helps you.
- Thank people — a lot.
- Give credit before you take credit.

These are low-effort habits that build high-trust relationships. And those relationships are your future power tools.



I like to think about the org the same way I think about system architecture.

Every team is a service. People are the interfaces. There are dependencies, data flows, and — let's be honest — bottlenecks.

As you move into a Staff-level role, you naturally start building a mental model of this. You start to know:
- Who owns what.
- Who's easy to work with.
- Who tends to block things — and why.
- Which Slack channels are where real work happens.

This mental map doesn't come from a doc — it comes from experience, repetition, and curiosity.

You build it by showing up, listening, asking questions, and noticing patterns.

And once you have that mental model, it gets way easier to navigate the org — to know who to pull in, how to approach them, and how to get things unstuck.

And that sets you up perfectly for what comes next: **influencing without authority**.

So — how do you actually get people to do things when you have no authority over them?

Maybe you've found an opportunity: the documentation is a mess, the onboarding flow is broken, or there's risk in how something is deployed. You see the fix — but the system isn't yours. The team isn't yours. What now?

This is where influence without authority becomes the most powerful tool you have. And the foundation of that influence? You've already been building it.
- You've done the work to know who owns what.
- You understand what matters to your peers — their pain points, their roadmap pressure, their OKRs.
- You've earned a reputation as someone who solves real problems and doesn't just throw tickets over the wall.

So now when you need something — you're not making a cold ask. You're connecting your goal to their goal. You're making the path easy. You're navigating the org like a routing protocol: finding the fastest, most reliable path to yes.

And if you need backup? You know which levers you can pull — peer advocates, senior engineers, or even a friendly manager who trusts your judgment.

That's what influence without authority looks like.
It's strategic. It's quiet. But it moves the system.
And if you want to dive deeper into this — this article I've linked here is one of the best I've found. I still refer people to it all the time.

TACTICS FOR BUILDING INFLUENCE

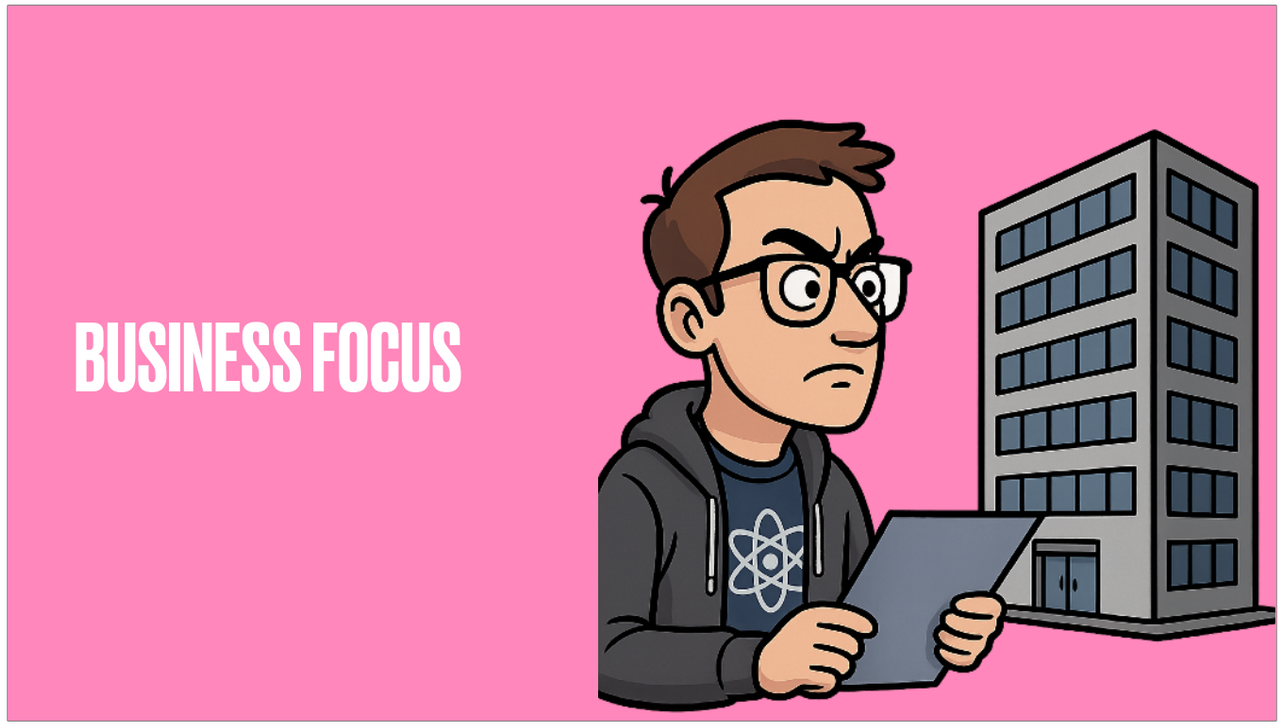
- Run the pre-meeting
- Trade visibility
- Ask stakeholders what success looks like

Okay, some actual tactics

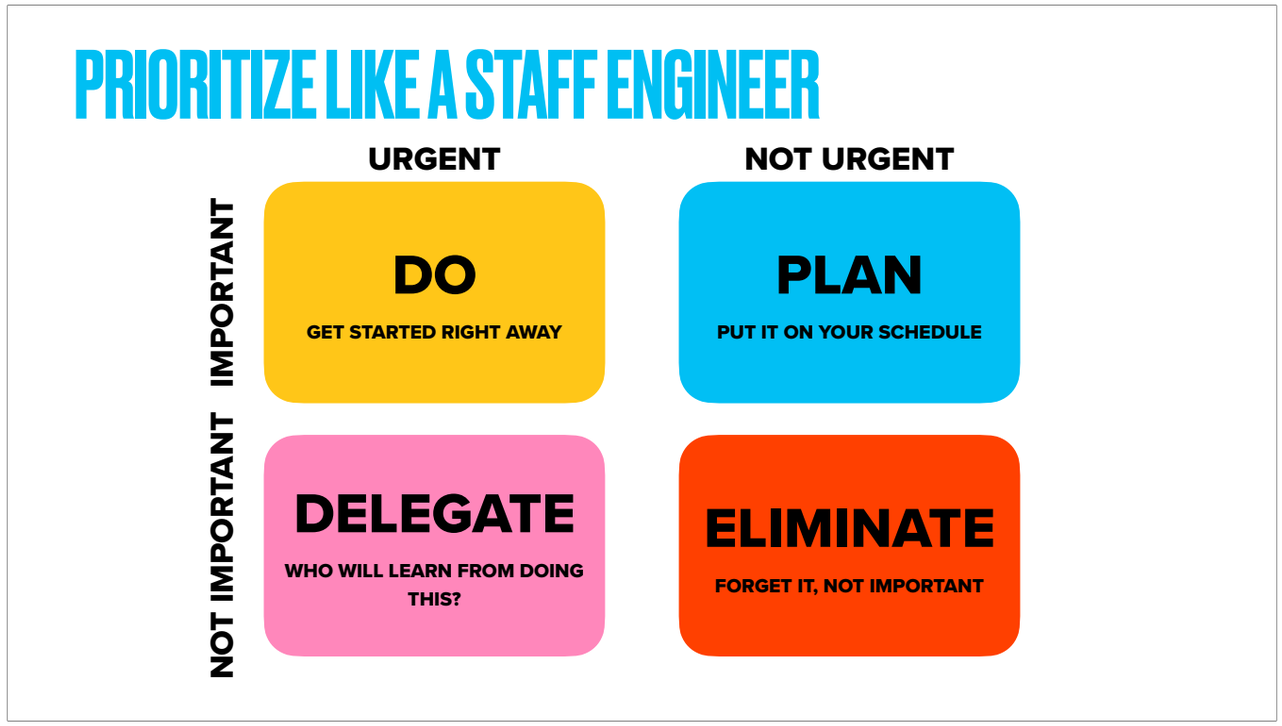
“Run the pre-meeting” – by pre-meeting I mean socializing your ideas with your key stakeholders before the meeting - when you have allies already in the bag, the others in the meeting are likely to follow

Trade visibility — promoting other people’s work will gain you allies

Ask Stakeholders: What does success look like to you? This shows that you’re aligned with their goals, not just your own. Instead of pushing your idea or solution, you're centering on their definition of value. This builds trust fast. - They see you as an ally, not a competitor for resources or attention.



This sounds so manager-y I almost feel a little sick. But a key attribute of a staff engineer is focusing on what is best for the business. It might be cool to build out a custom tool to do something, but you need to have a real justification to do so. If there is an off the shelf solution that works fine for what you need, why are you suggesting to build one? And sometimes it all comes down to reframing the problem you want to solve to align with the business needs. Lets say you want to migrate to a new mdm because you dislike the UI of your current one. That’s not a business need. The new one is more expensive, so you can’t use that as a reason. But if you spent a little time calculating how long it takes to make the average change and then multiply that by how often you make changes, then you’ve got a compelling business case. Another example: lets say there is a bug in macOS that stops a small portion of your company from working, but that small portion is working on something vital for your next product launch? Obviously you want to be secure, so perhaps you allow the fleet to install the update, but ask those few users to not update. Or if you look at it from a business perspective, you can’t take the risk of these users not being able to work, and perhaps you can’t reliably identify them, so you decide to hold back the update from the entire fleet until the product launch is over.



One of the hardest shifts at Staff level is figuring out what **not** to do. You can't be everywhere. You can't fix everything.

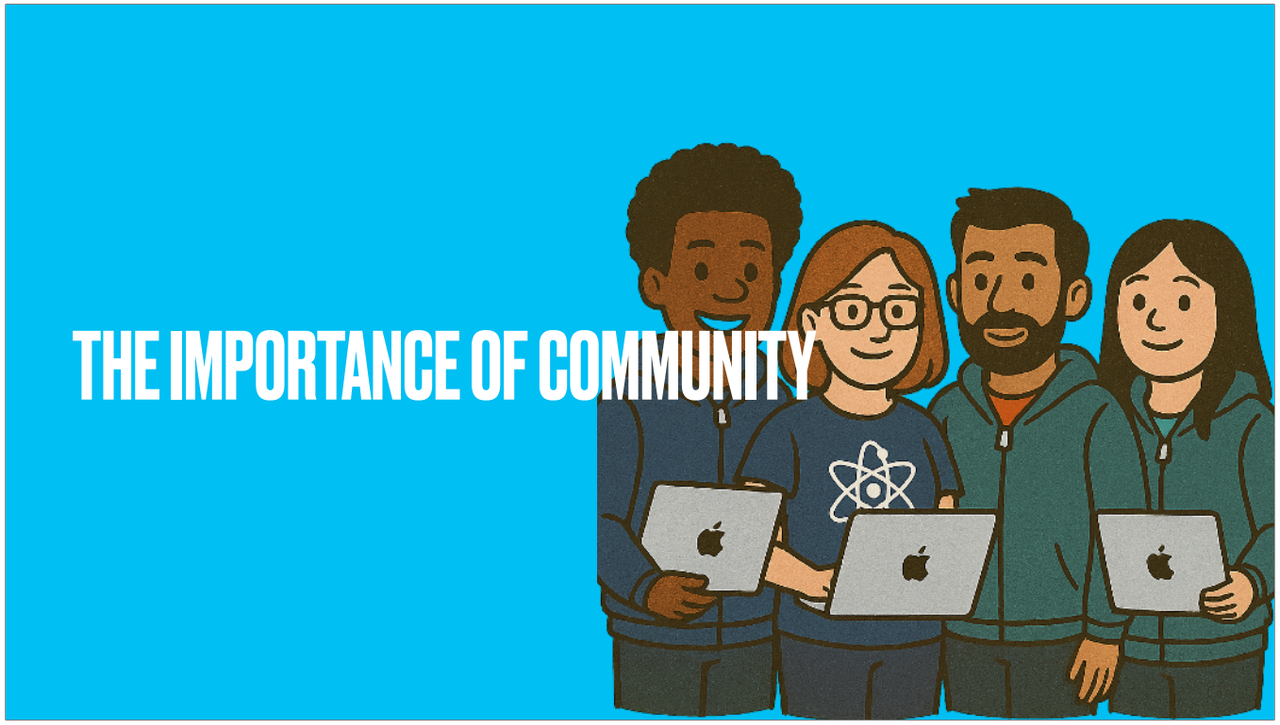
This is where I come back to the **Eisenhower Matrix** — a really simple framework that helps you decide where your energy should go.

- Urgent and important? You're probably on the hook.
- Important but not urgent? Put it to one side. That's where your highest-leverage work lives, think about it when you've got more time.
- Urgent but not important? That's a growth opportunity for someone else. Delegate it.
- Neither? In the words of Elsa, let it go. Seriously. Or at worst, write a script and never think about it again.

Staff Engineers aren't just builders — we're force multipliers.
Part of your job is to create space — for focus, for strategy, for coaching others. This framework helps you do that without guilt.

This isn’t just for your time — it’s how you level up your team.
Junior and mid-level engineers are often stuck in that urgent-but-not-important quadrant.
Pull them up. Delegate wisely. Coach them through it. That’s how they grow, and how **you** scale.

The more intentional you are about this, the more room you have to operate at the level your org really needs you.



So far we've talked about technical skill, communication, leadership, and business focus. But there's another piece that often gets overlooked — and that's community.

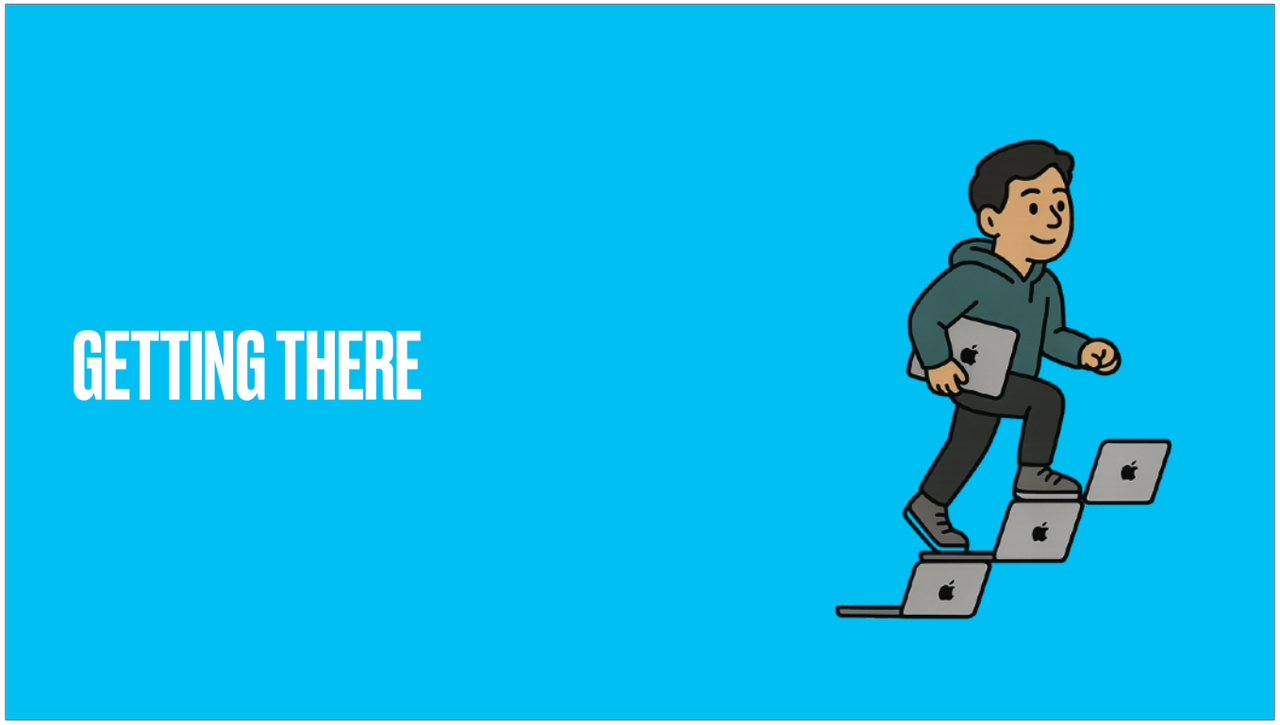
At Staff level, your reputation matters. Not just inside your company, but outside it too. Why? Because your visibility adds weight to your ideas. Because being part of a broader professional network gives you insight, inspiration, and allies. Because sometimes the best solutions come from people you've never met — until you connect with them at a conference or online maybe.

But also — because the community needs leadership. The next generation of Mac Admins needs role models, mentors, maintainers, speakers. Staff-level engineers give back. They contribute. They help raise the bar not just for themselves, but for everyone else too.

And that ties directly into this: build your brand. This doesn’t mean becoming an “influencer.” It means sharing what you know. Owning your perspective. Being visible in your domain.

That might look like:
- Speaking at a conference like this one.
- Writing a blog post or case study.
- Sharing tools or code you’ve built.
- Going on a podcast or joining a community meetup.

These things show the world — and your company — that you know your stuff. That you care about the craft. That you're pushing the industry forward. And when your name carries weight outside the org, it carries more weight inside too. So build your brand. People can’t advocate for you if they don’t know what you do.



So let’s say all of this is resonating with you.

You’re technically strong, you care about the craft, you’ve started thinking about longer-term problems — maybe you’re already doing some Staff-level work without the title.

The question is: How do you make the leap?

There’s no checklist. No magic formula. But there are a few things that helped me get there — and I’ve seen them work for others too.

One of the most valuable things I did was ask to be invited to meetings that were, frankly, above my pay grade. I didn’t go to speak. I went to listen.

When you’re in the room where decisions get made — planning meetings, roadmap reviews, cross-functional working groups — you start to understand how leaders think. You learn what matters to them. What trade-offs they’re making. And once you understand that, you can start to frame your own work in ways that land with them. You see how a patching strategy connects to risk mitigation. How automation connects to operational efficiency. How device health ties into compliance and business continuity.

You also start spotting gaps — problems you’re uniquely positioned to solve — and you’ll know how to pitch them. If you’re not sure how to ask when you want to get into these meetings, keep it simple:

“I’m trying to understand how decisions get made so I can better align my work with org goals. Would it be okay if I sat in on this?”

Eventually, you won’t have to ask. You’ll get the invite. Because people will see that you think beyond your role. That you belong in the room.

Another important step if you want to grow into a Staff Engineer role: start asking leaders what's on their mind. I mean literally — go to a manager, a director, someone adjacent to your org and ask:

- "What's keeping you up at night?"
- "What are you worried about in the next 6 to 12 months?"
- "Where do you see risk, friction, or things just quietly going sideways?"

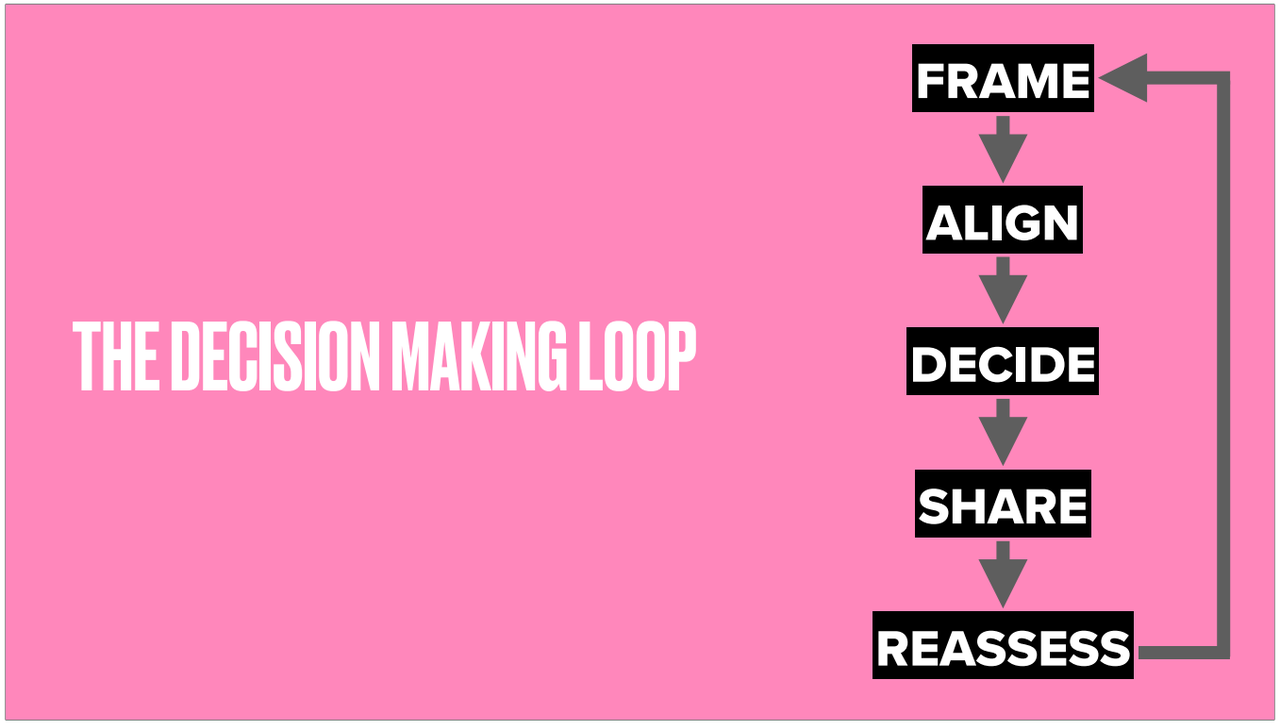
Why? Because Staff Engineers don't just solve technical problems — they solve organizational ones. And the biggest, most strategic problems? They're usually not written down in Jira. They're floating around in an exec's mental backlog — half-formed worries about scale, reliability, security.

When you ask these questions, three powerful things happen:
- You get real insight into the bigger picture.
- You build trust — you're showing that you care about more than just your own code or your team's OKRs.
- And sometimes, you get handed a problem no one else owns yet — and you get to shape the solution from scratch.

That's how high-impact projects start. That's how you get noticed. That's how you earn sponsors who will go to bat for your promotion.

Now — quick show of hands: who here has actually asked a VP or Director what's keeping them up at night? …Not many? That's okay. That just means you've got your next growth experiment ready to go.

Don’t wait for perfect clarity or permission. Start with a conversation. Start with curiosity. Start by asking what’s keeping them up at night.



As a staff plus engineer, it’s no longer enough to just make good technical choices. You have to make the right decisions for the business, often in complex, ambiguous situations. One pattern I’ve found helpful is thinking in terms of a **Decision-Making Loop** — a repeatable cycle that gives structure to how you lead your team in making decisions.

First - Frame the problem - Don’t jump to solution mode. Start by asking: What’s really at stake? Align on whether this is a technical problem, a people problem, a prioritization problem.
For example: A flaky internal tool might not need a rewrite — maybe it needs a clearer owner.

Second - Align stakeholders. Once you’ve framed the issue, bring in the right voices early. This is where influence starts: you’re making the problem shared. A great tactic for doing this is: Asking “What does success look like to you?” to uncover silent criteria.

Third - Decide and document - Staff engineers don’t just make decisions — they write them down. Clarity builds alignment. Even a short Slack post: “Here’s what we’re doing, and why.” Is great

Next - Share & socialize- You’ve made the decision — now help others understand it. Build visibility in a way that doesn’t feel self-promotional: link to a doc, recap in a channel, offer to present at a team meeting.

Finally - Reassess - Not all decisions age well. Staff engineers check back: “Is this still working? Should we revisit?”

So, finally—you need to learn to be a leader. And let’s be honest: for a lot of us in this field, that doesn’t come naturally. It didn’t for me. Earlier in my career, I was all about getting things done the right way. I was focused, opinionated, and… a bit of an asshole. I thought being right was the goal. But I wasn’t thinking about how I made people feel.

In one of my early performance reviews, my manager included a bit of feedback from our support team. They said that they were too intimidated to ask me questions directly. Instead, they’d ask them via the Mac Admins Slack—because it felt safer. That stung. But it was a turning point. I realized being technically good wasn’t enough. At the Staff level, you’re not just solving problems—you’re shaping how other people solve problems. You’re not just executing work—you’re influencing the direction of the work. And you can’t do any of that effectively if people don’t want to engage with you. I had to learn to listen. To explain. To empathize. To support others and not just correct them. That’s what leadership actually is: making space for others, not just taking space for yourself.

And I’ll leave you with this quote—something my manager said to me when he delivered that same performance review. It’s stuck with me ever since: “You cannot be a leader without followers.” That might sound obvious, but it’s incredibly important. You can’t be a leader just because you’ve got the job title, or the experience, or the strongest opinions in the room.

You have to earn trust. You have to make people want to follow you—not because they have to, but because they believe in you. Because they believe that them doing work on the things **you** think are important is worth **their** time. And that means you can’t be an asshole. You have to be the kind of person people want to work with, want to listen to, and want to build something with.

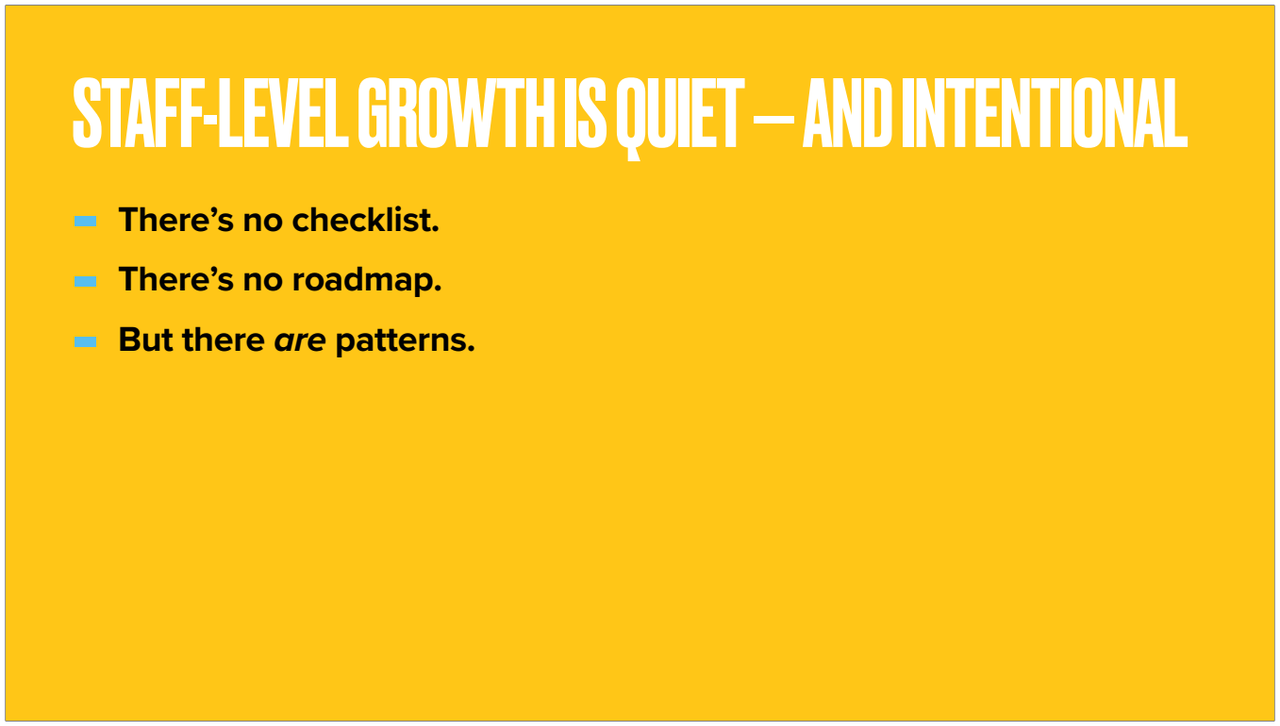
That’s leadership. And that’s the part that turns a great individual contributor into a Staff Engineer.

Alright — we've covered a lot.

From what Staff actually means, to how to build influence, to how to think more like a leader — even when your title hasn't caught up yet.

So let's take a breath.

I'm going to leave you with a quick recap, a few traps to watch out for, and a reminder that you're probably a lot closer than you think.

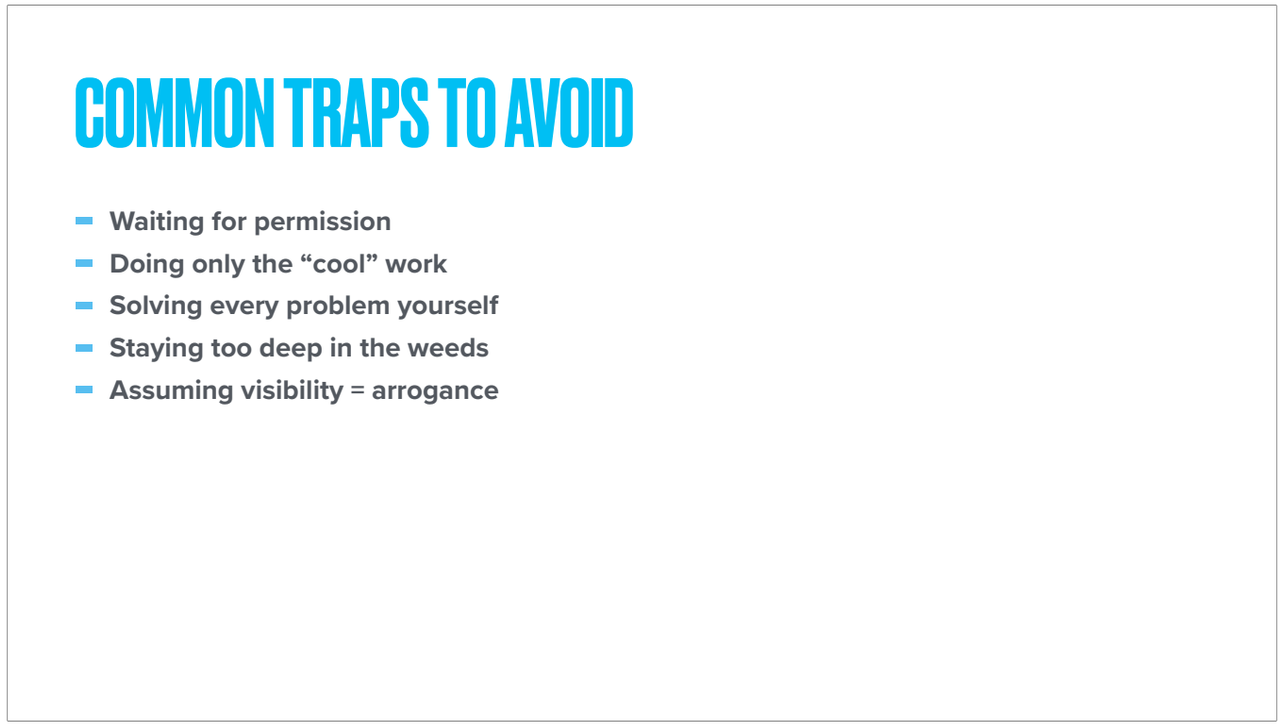


Most people don’t get promoted to Staff because they were the loudest, or because someone handed them a project with a bow on it.

They get there by quietly doing the hard, strategic, connective work that moves the org forward.

Staff-level growth doesn’t usually look glamorous. It looks like being in the room early. Fixing the problem no one else has named yet. Getting the right people aligned.

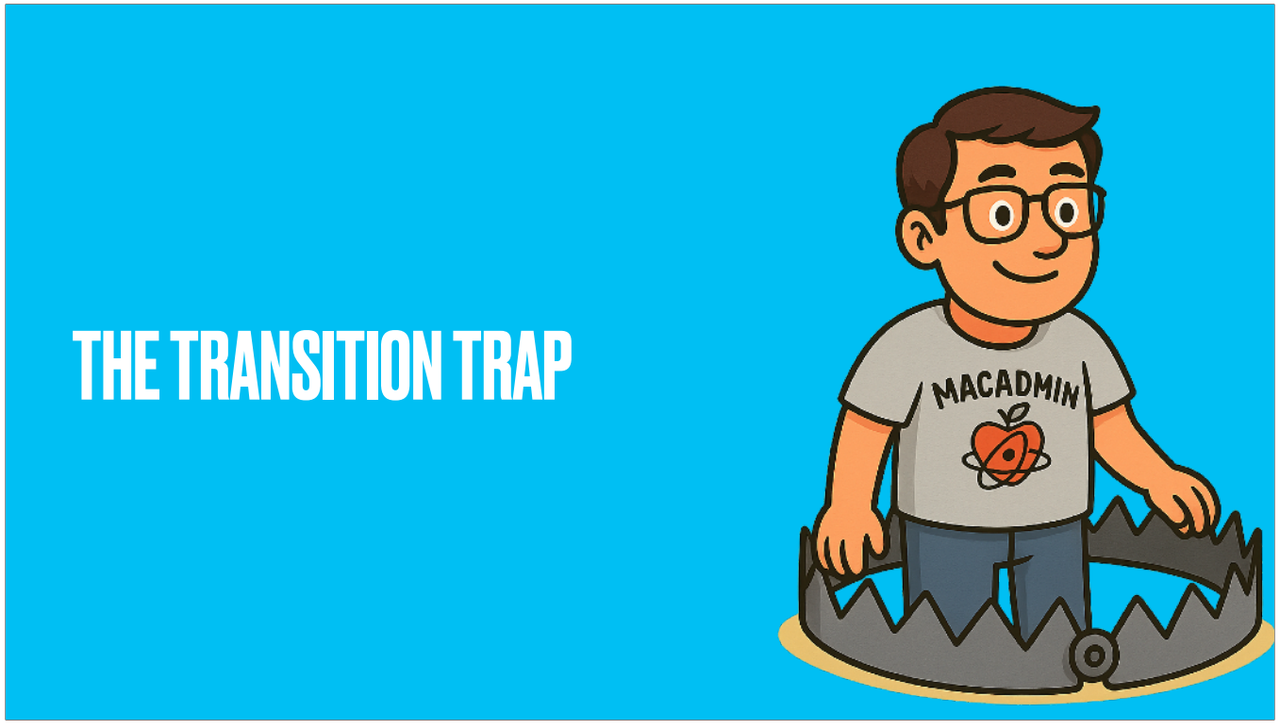
It’s quiet. It’s intentional. But it gets noticed.



If you’re aiming for Staff, you’ll run into traps. I hit a lot of these myself.

- Waiting for permission? You’ll miss the opportunity.
- Only doing the cool work? You’ll never build trust.
- Solving everything yourself? You’ll burn out, and your team won’t grow.
- Staying too deep in the weeds? You’ll miss the big picture.
- And thinking “if I’m visible, I’m bragging”? No — visibility is how people know what you bring to the table.

Staff Engineers don’t just do work. They shape work. That means stepping up — and stepping back — at the right moments.



So let’s talk about something I think a lot of senior ICs run into — and honestly, something I ran into myself. You’re doing great work. You’re respected. Your code is flying out of the door. You’re often the one people turn to when something breaks. But despite all that, you’re… stuck. Not moving forward. Not quite hitting Staff.

This is what I call the **Transition Trap** — the gap between being a really solid senior engineer and actually operating like a Staff+ engineer. And the reason it’s so tricky is that… nothing is wrong. You're not underperforming. You're not failing. **You're just still playing the game the same way** — but expecting a different outcome.

What I’ve seen — and felt — is that the behaviors that made you successful at Senior can start holding you back.

Like:
You’re still rewarded for execution, so you keep shipping more and more — instead of stepping back and shaping what gets built.
Or you feel a sense of pride in taking on the hard stuff — solving the gnarliest bug, writing the best code — but you’re not enabling anyone else.
Or maybe you’ve become absolutely essential to your team… but invisible to the org.

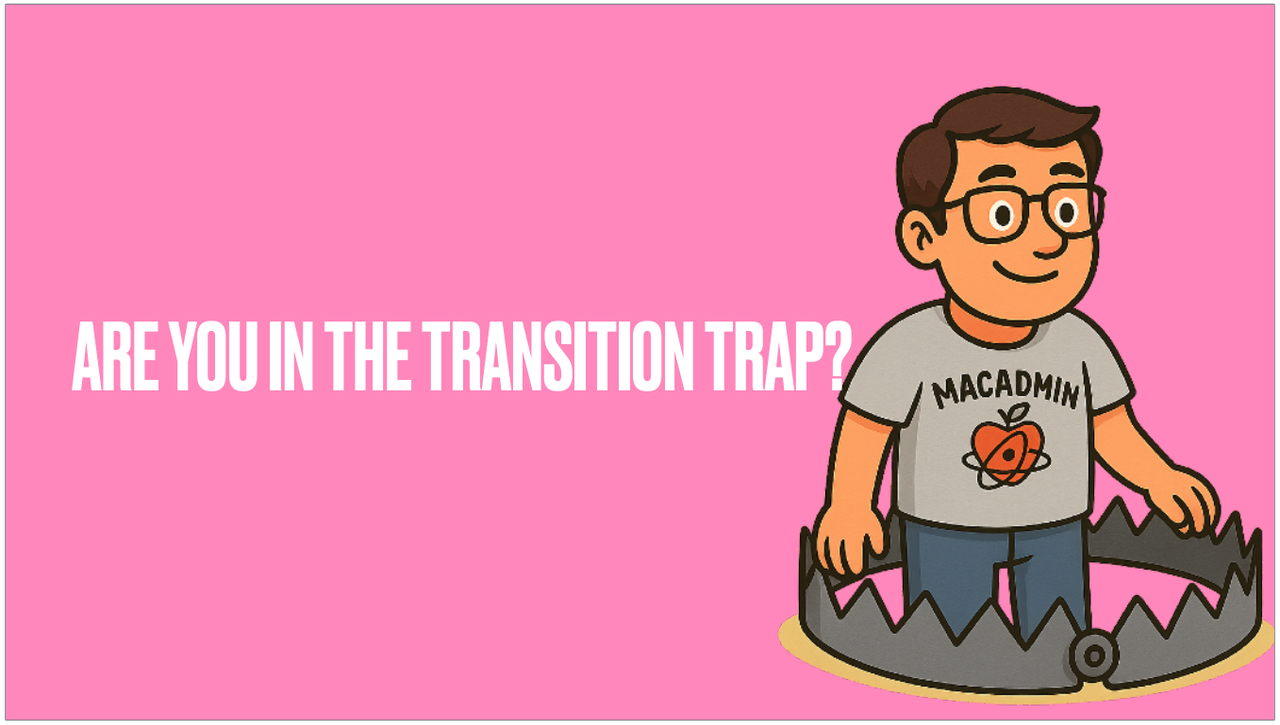
And I get it. I’ve been that person. That’s comfortable. That’s familiar.

But if you’re trying to move forward, you have to deliberately switch gears.
And that starts with asking: “What do Staff Engineers actually do differently?”

They start from outcomes — not tickets. They let go of being the smartest and focus on being the most enabling.
They zoom out — from their team to the broader org. And they learn to say no — to low-leverage work, to work that doesn’t move the strategy forward.

And most of all — they stop trying to prove themselves the same way they did before.

Because moving to Staff isn't about doing more of what you're already good at.
It's about changing how you operate.
That's the transition. And if you can make that shift — you're already on your way.



So if any of that last slide resonated, you might be wondering — **am I in this transition trap**?

Here are a few signs I look for — both in myself and in others I've mentored.

First: You're the go-to person for execution. You're trusted to deliver. But you're not part of the early conversations — the shaping ones. That's a sign.

Or: You're constantly solving problems yourself. And that feels productive — but Staff engineers build systems, unblock people, or prevent those problems in the first place.

Another one: Your relationships are deep within your immediate team… but thin across the org. If you want to operate at Staff level, your impact has to spill out beyond your function.

Maybe the most common: you are extremely busy. You're slammed. But when you zoom out, it's hard to tell if what you're doing is actually moving the needle.

And finally: you're waiting. You're waiting for someone to notice, to nominate you, to give you permission to level up.
But the real shift often starts when you stop waiting — and start acting like you're already in the role.

Staff engineers don't ask "Can I?" — they start doing the work, and then the title catches up.

So if you see yourself in any of these patterns — that's not a failure. That's awareness. And awareness is the first step out of the trap.

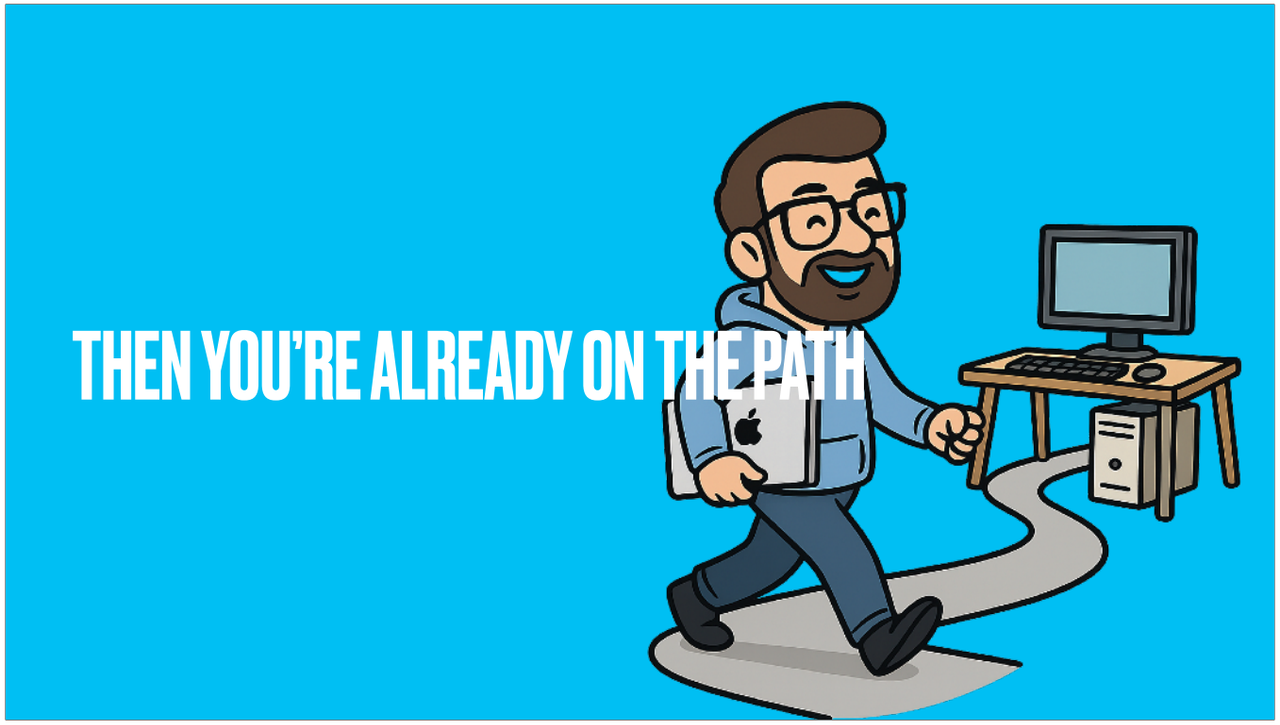
ARE YOU ALREADY...

- Mentoring?
- Shaping direction?
- Seeing the gaps?
- Thinking long-term?
- Building trust across teams?

If anything I've said today sounded familiar — that's not an accident.

A lot of you are already doing pieces of this work. You're mentoring. You're thinking bigger. You're pushing your team forward.

Maybe you just needed the language for it. Or the confidence to own it.



If that's you — keep going. You're already on the path to Staff engineer.

Okay, you've all sat there long enough - time to put you all to work.
I'm going to ask you a few questions — and if you're comfortable, I'd love to hear some answers out loud. Please stick your hand up and wait for the microphone to come your way
We're going to dive into where we are spending our time, and whether we are spending our time on the right things
Don't overthink it — just give me the first example that comes to mind.
No judgment here — we've all been in the trap of doing work that feels urgent but might not be strategic.

“What’s one thing you did last week that made a real difference to your org?”

Follow-ups:

“Why do you think it made a difference?”
“Did that work come from your team’s backlog — or did you create space to do it?”
“Who else benefited from that work?”
“Would your org have missed it if you hadn’t done it?”
“Did it scale beyond your team — docs, tooling, process?”

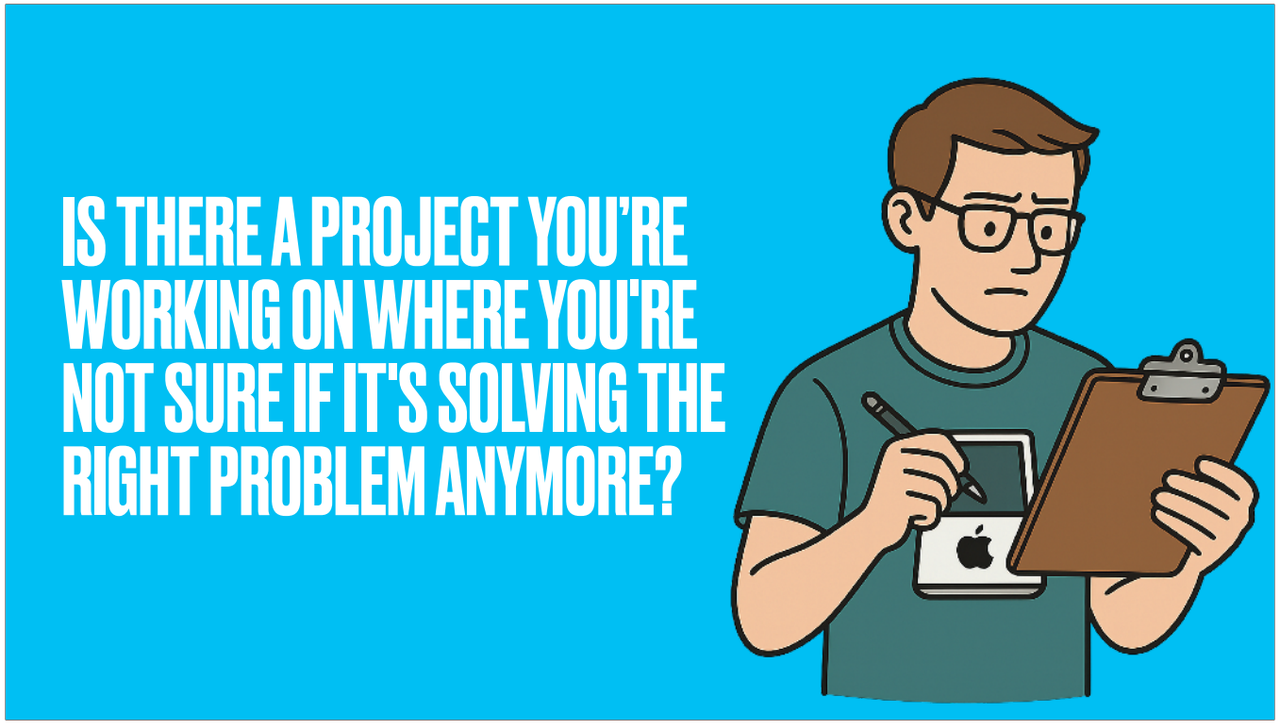
If no answers come quickly:
“No pressure — just think about something that’s still having impact today.”
“Was there a conversation you had that shifted a decision?”

“What’s one task you’re doing right now that someone else could learn from doing?”

Follow-ups:

“What’s stopped you from handing that off?”
“Is the blocker trust, time to teach, or just habit?”
“What would happen if you did delegate it?”
“Is there someone newer on the team who’d grow by taking this on?”
“Would writing it down once save you from doing it again next week?”

If the audience hesitates:
“Think about anything you’ve done 3+ times in the last month — is it truly the best use of your time?”



“Is there a project you’re working on where you're not sure if it's solving the right problem anymore?”

Follow-ups:

“How did this project get started — and has the context changed?”
“What was the original goal, and is that still the priority?”
“If someone asked, ‘Why are we doing this?’, how confident would your answer be?”
“Is there a faster way to get the same outcome?”
“What would you do differently if you were starting fresh today?”

If people are quiet:
“It’s okay to say yes — sometimes we’re halfway through a build and realize the need shifted.”

“What’s one meeting you attend that you suspect could disappear… and no one would notice?”

Follow-ups:

“Why do you keep going?”
“What would need to change to make that meeting worth your time?”
“Could you send someone else, or ask for notes?”
“If you stopped attending, what would you miss?”
“Would replacing it with a doc or async update be better?”

“If you had 20% more time each week — what would you actually focus on?”

Basically - what are you putting off because of noise?

Follow ups:

- What’s stopping you from doing that now?
- What could you stop doing to make space for that?
- Would that work raise your visibility or amplify others?
- Is this something only you can do — or something you’re just used to doing?
- If you did that for 3 months, what would be different for your team or org?

START ACTING LIKE A STAFF - NEXT WEEK

- Shadow a cross functional meeting
  - Don’t talk. Just observe how decisions get made
- Ask a leader what’s keeping them up at night
  - Use their worries to shape your next big idea
- Question one habitual process on your team
  - Ask: is this solving the *current* problem?
- Introduce yourself to someone in another org
  - Build the relationship before you need it
- Share something in public
  - A doc, a Slack post, a useful bug fix. Visibility compounds.

That was a lot. But I want to leave you with something practical.

Staff isn’t a title you wait for. It’s a role you grow into by showing up differently.

Here are five things you can try — literally next week when you get back from this conference — that build the habits, visibility, and judgment of a Staff Engineer.

You don’t need permission. You don’t need perfect clarity. You just need to start.

Pick one, try it, and see what changes.

So that’s it - we are hiring a security person with a focus on endpoint, and a global head of IT support in the US and a couple of senior systems engineer roles in India - if you are interested in learning more about any of those roles please chat with me.

I have coffee chats with people twice a week - coffee chats are one offs where we can chat about whatever you would like — if you would like to schedule one, please hit graham.at/coffeechat

You can find me on these places - I have a rarely updated website and you can email me at my very cryptic and hard to guess email address. And if you found this talk useful in any way, please consider donating to the Movember foundation and help save some lives of people you might very well be sitting next to.

And now we can take some questions