GRAHAM GILBERT / MACDEVOPS:YVR 2025

THE PATH TO STAFF ENGINEER AND BEYOND

STAYING ON THE IC TRAIN

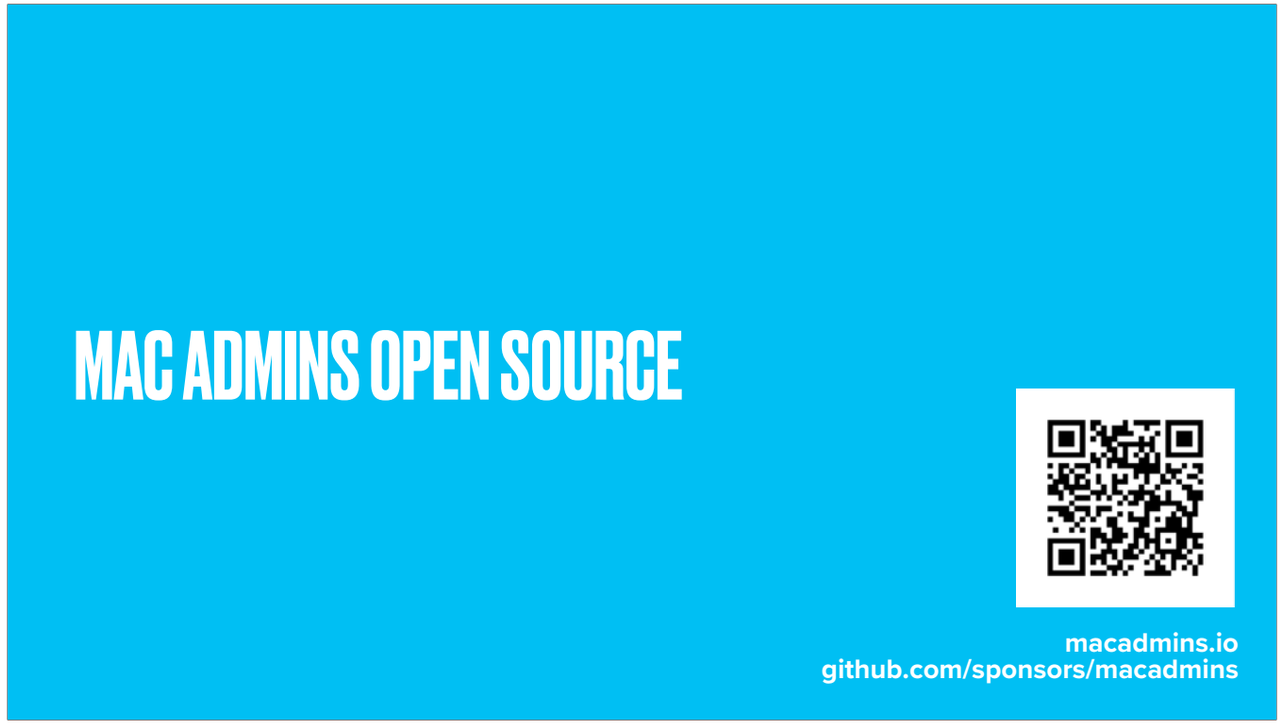
The views expressed in this talk are my own and do not necessarily reflect those of my employer

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

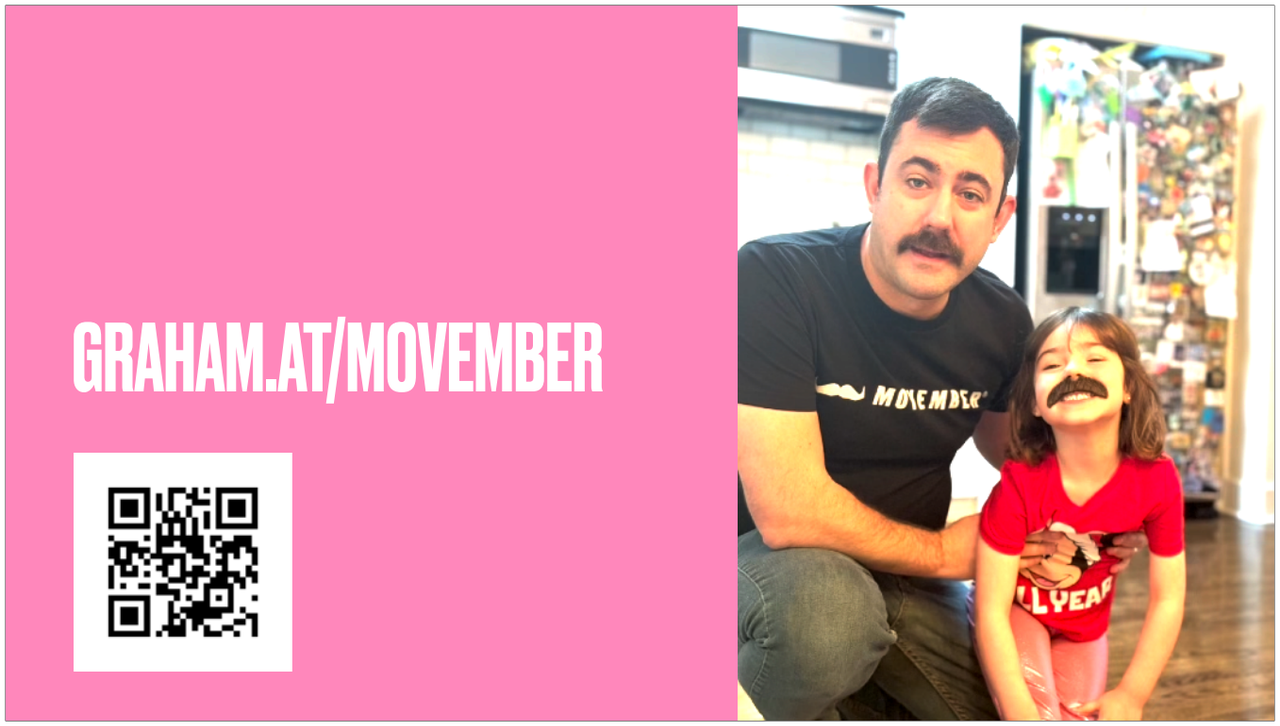
HELLO!

- 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.



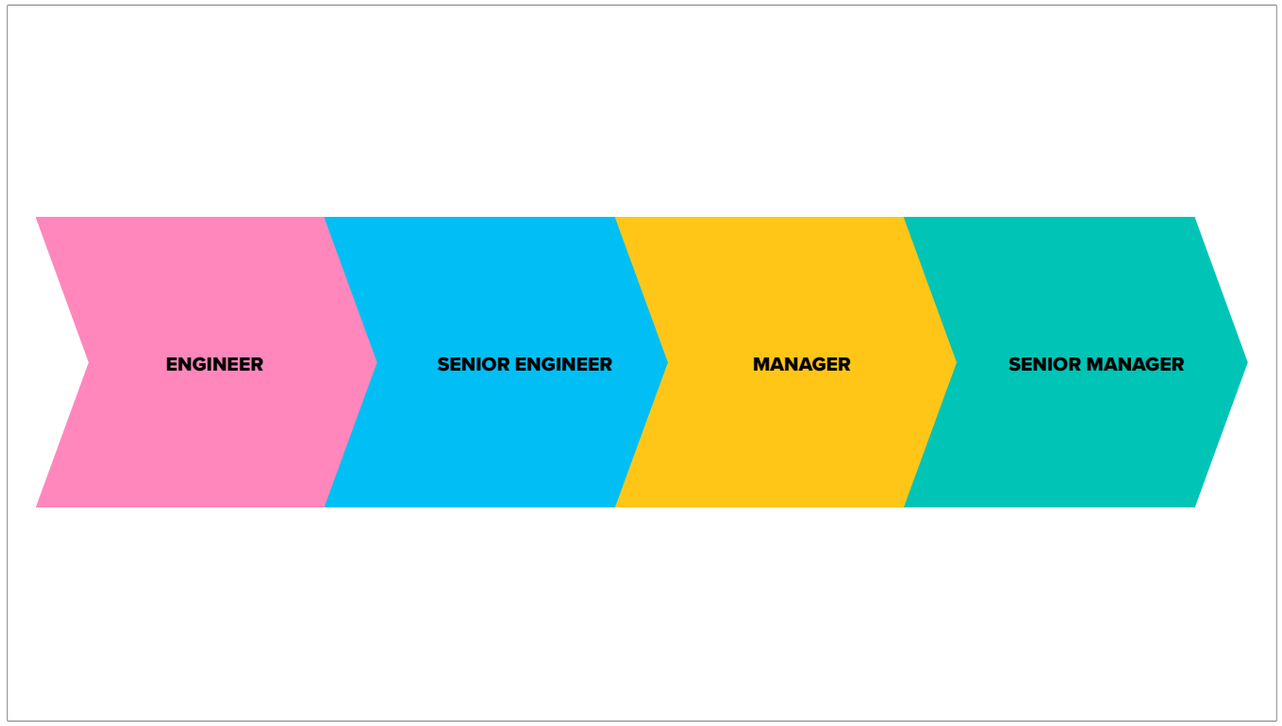
I am on the board for Mac admins open source, and we do things like supporting the SOFA feed and signing munki. Running Mac admins open source isn’t free. We have a not inconsiderable cost for the hosting and bandwidth for Sofa in particular. If you use Sofa, and find it useful, or find having a codesigned Munki or osquery extension useful, or even use any of our osquery tables, 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. So if you find this talk useful, please 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?

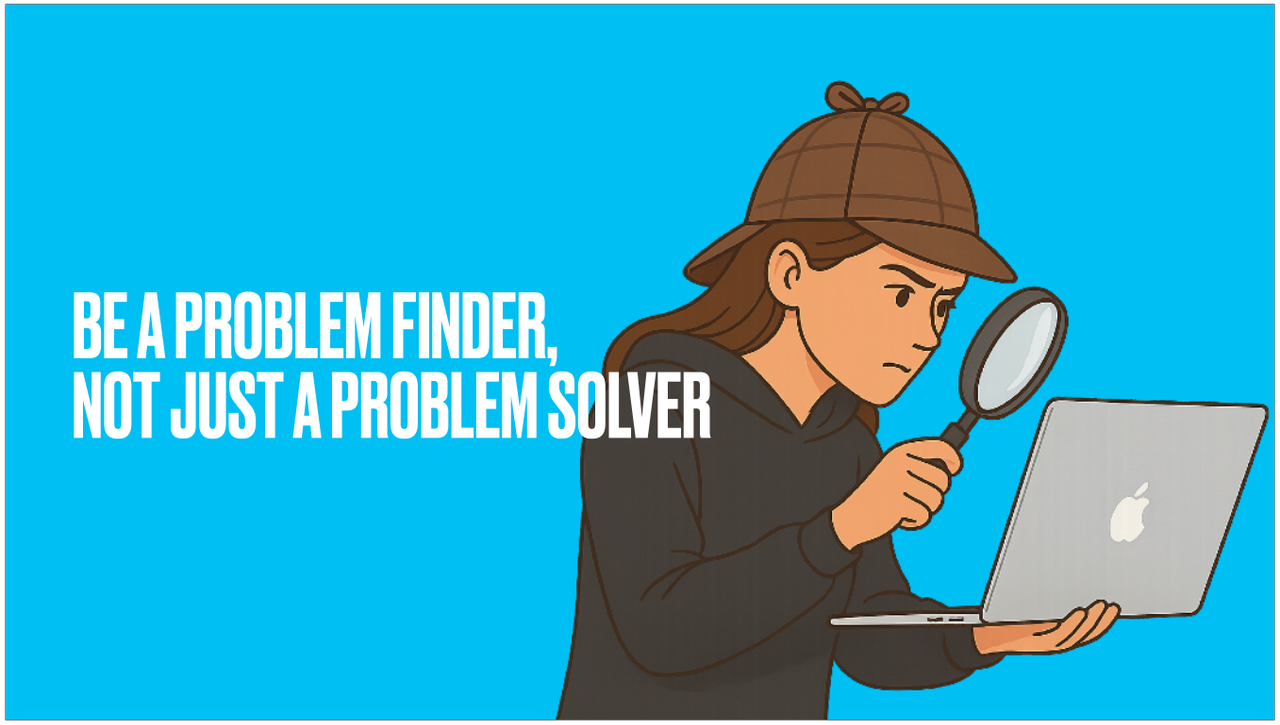
STAFF ENGINEER

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.

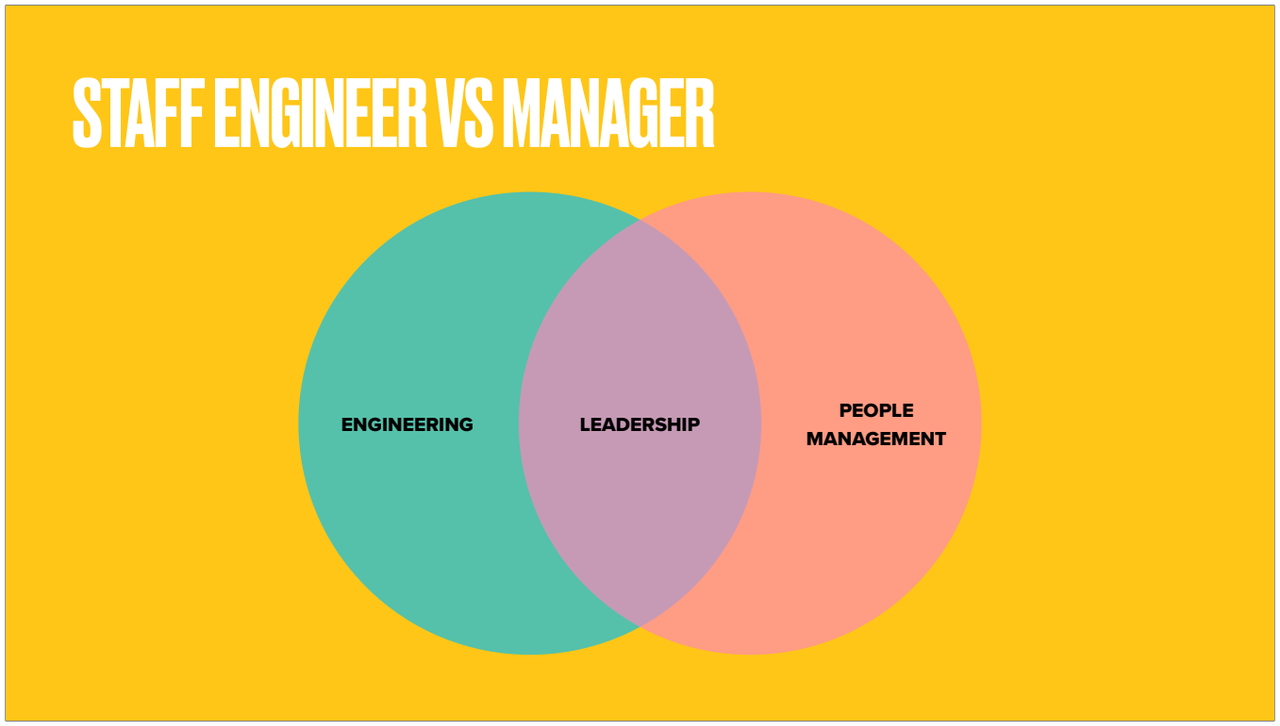
The best Staff Engineers leave behind a wake of better engineers, smoother systems, and smarter decisions.

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.”

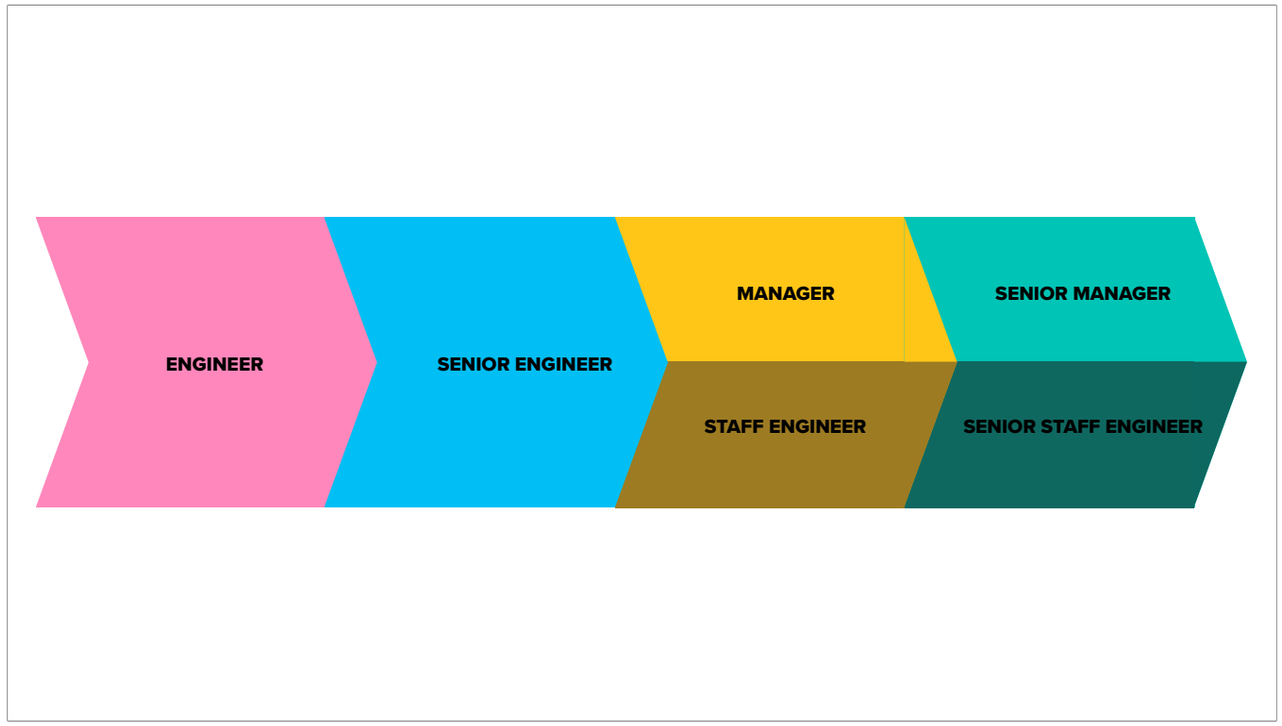
And… yeah. You’re not wrong.



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.

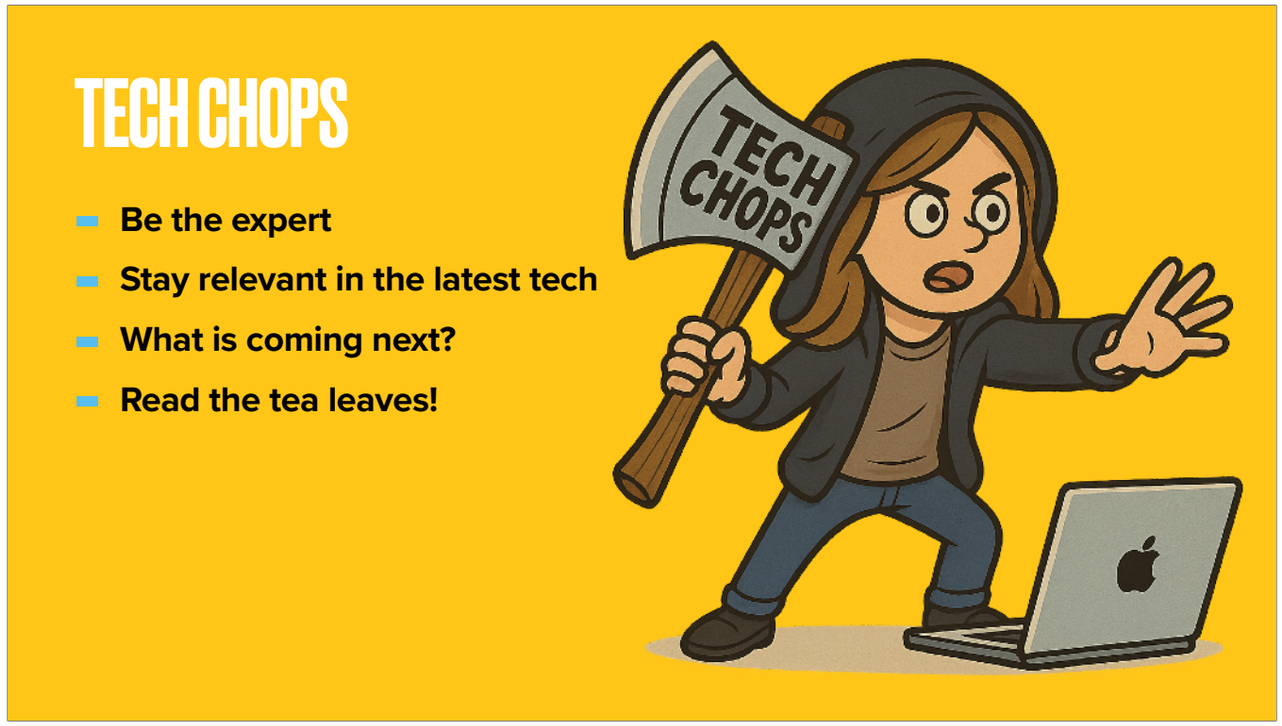
OKAY, WHAT DO I NEED TO BE A STAFF ENGINEER?

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 often vague documentation.

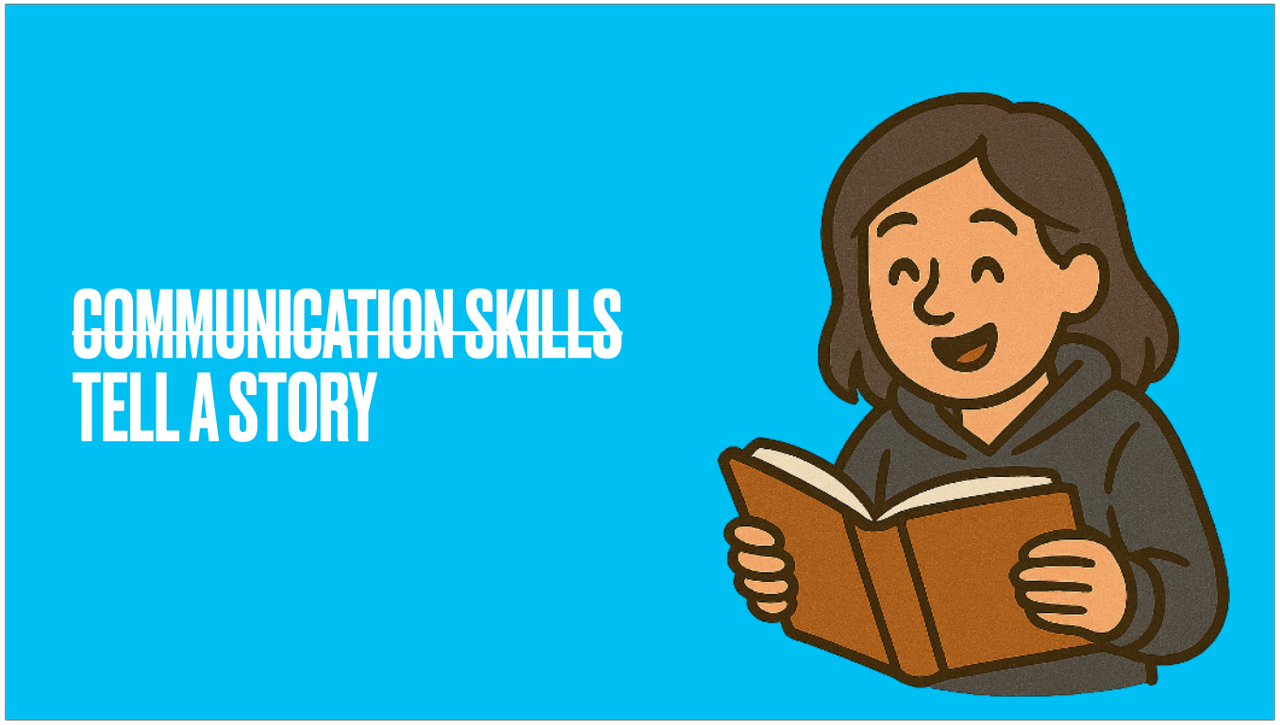
We work in an ecosystem where we don't get roadmaps. So part of your job is to read the tea leaves. Being technically strong is what gives you the platform to influence — because people trust engineers who know what they're talking about.

COMMUNICATION SKILLS

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

Now you need to communicate it.

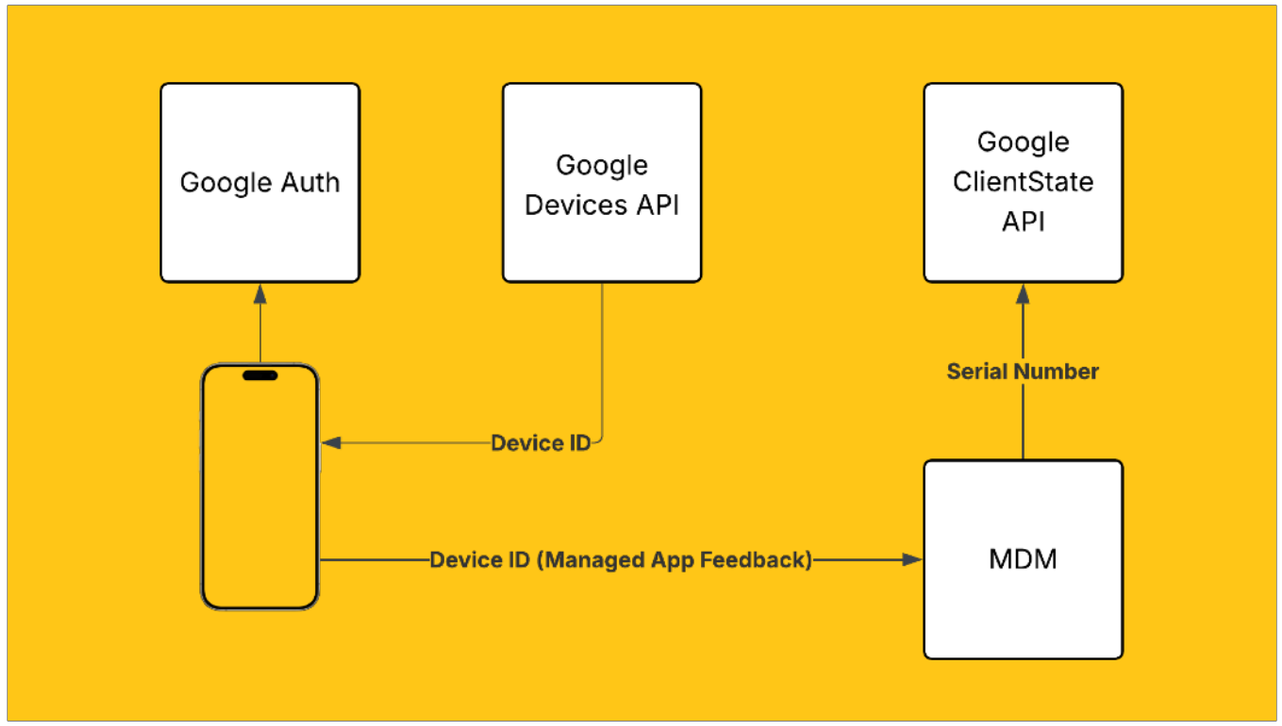
And I don't just mean writing clear documentation or speaking up in meetings — though both are important.



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.

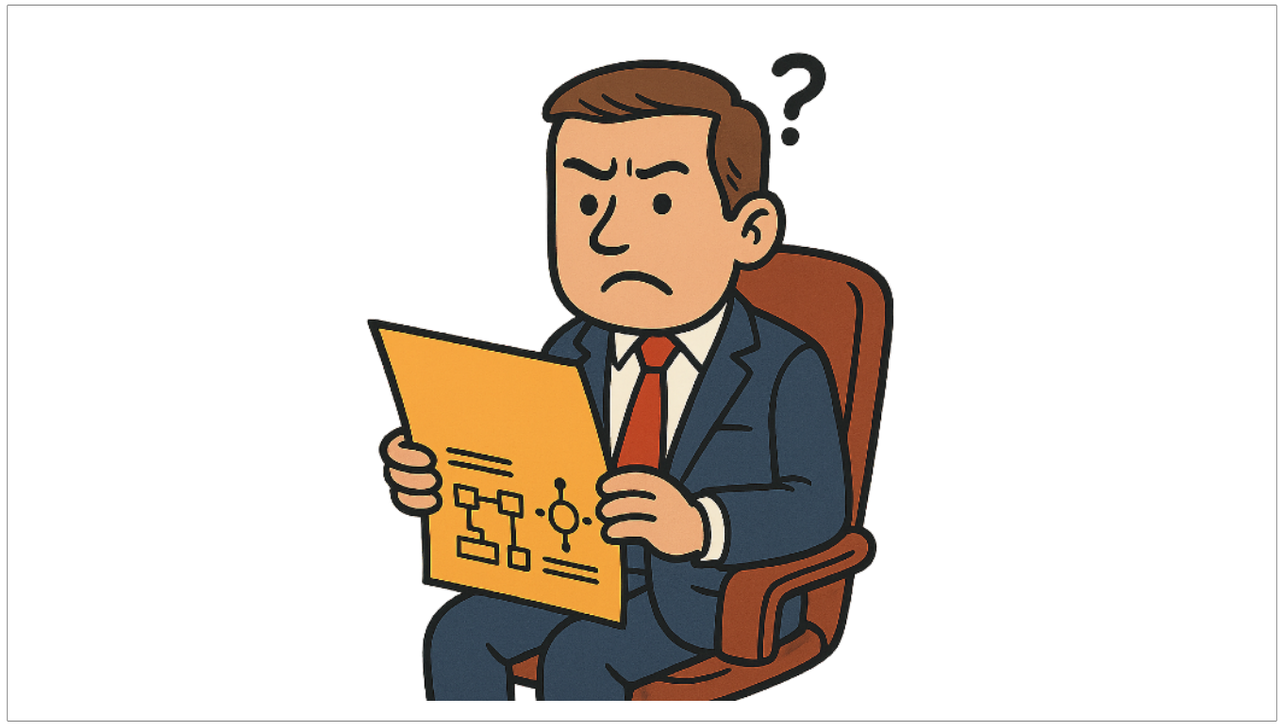
Same problem, different message.



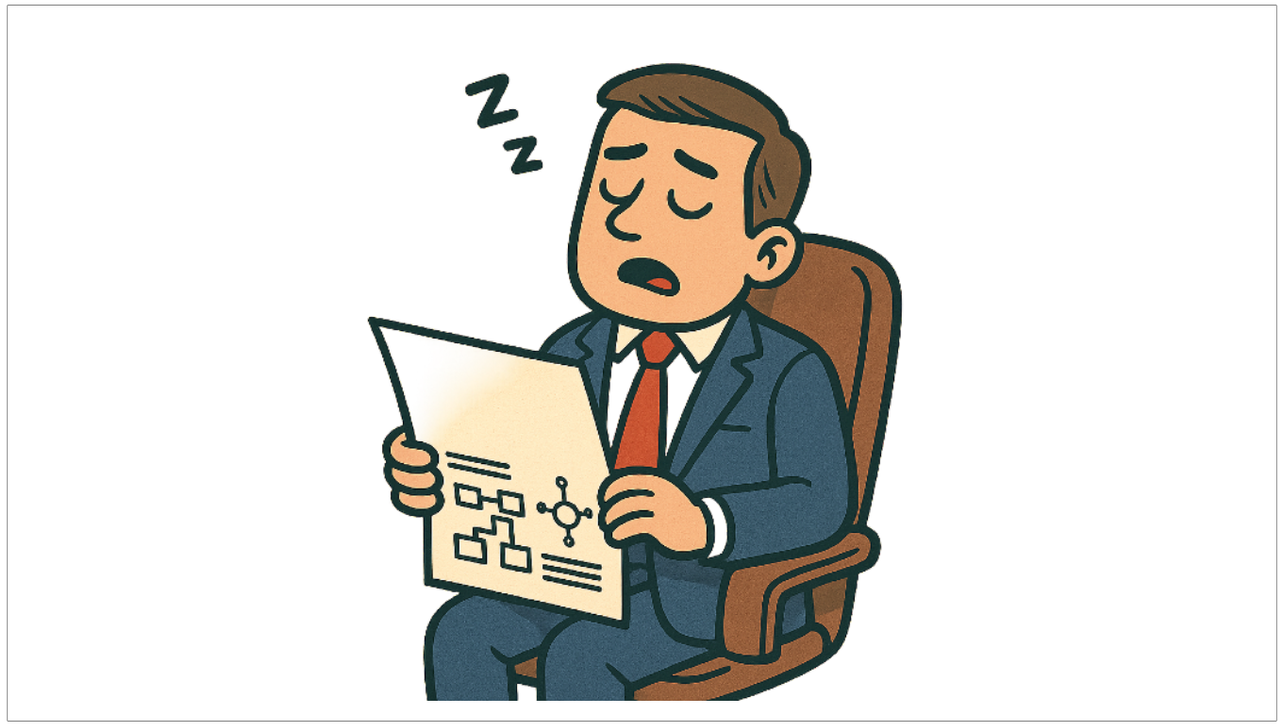
Let’s say you’re proposing an iOS app to communicate with end users and integrate with Google context-aware access. You’ve got the design doc, the architecture, the security model.

Great — that’ll make sense to your engineers.

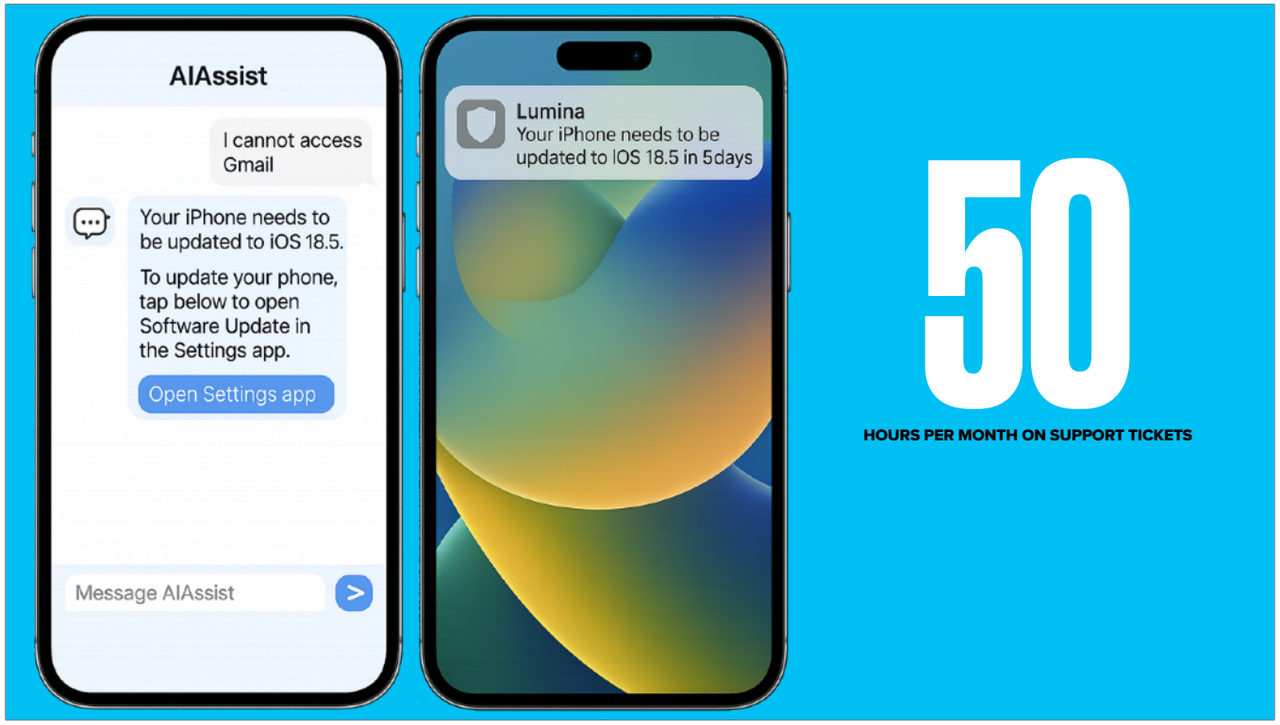
But if you send that same doc to an exec?



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.

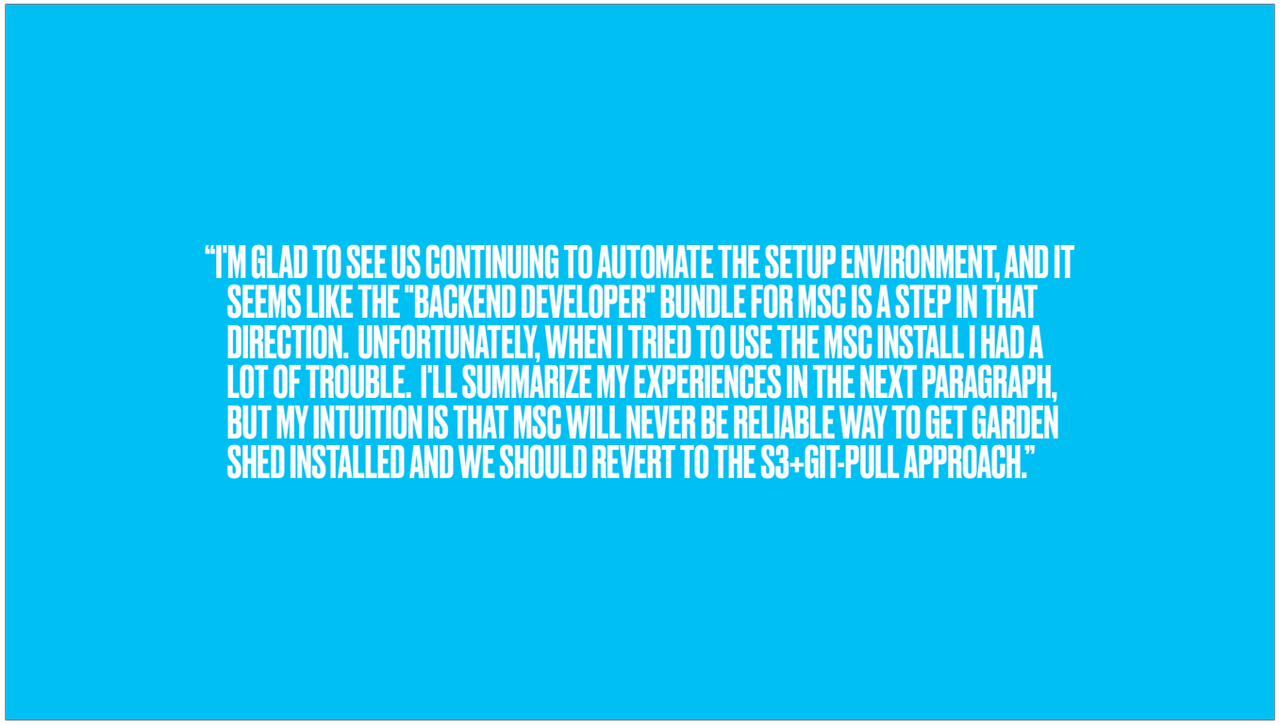
Now, let’s talk about opportunity. When you’re early in your career, you’re mostly handed projects. Someone tells you what needs doing, and you do it.

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. Sometimes it shows up as a mess — political, unscoped, underfunded. And your job is to see the potential in it.

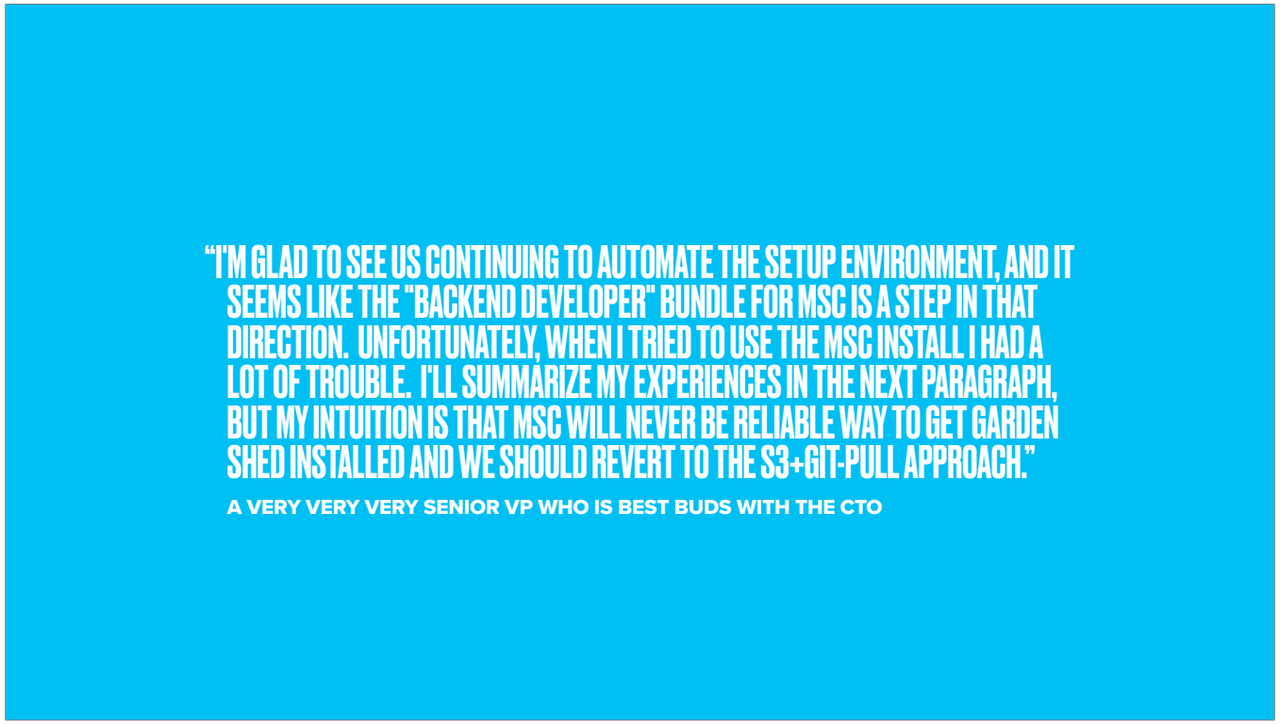
PROJECT: DEVELOPER SETUP AUTOMATION

- 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
- Profit?

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.



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_.

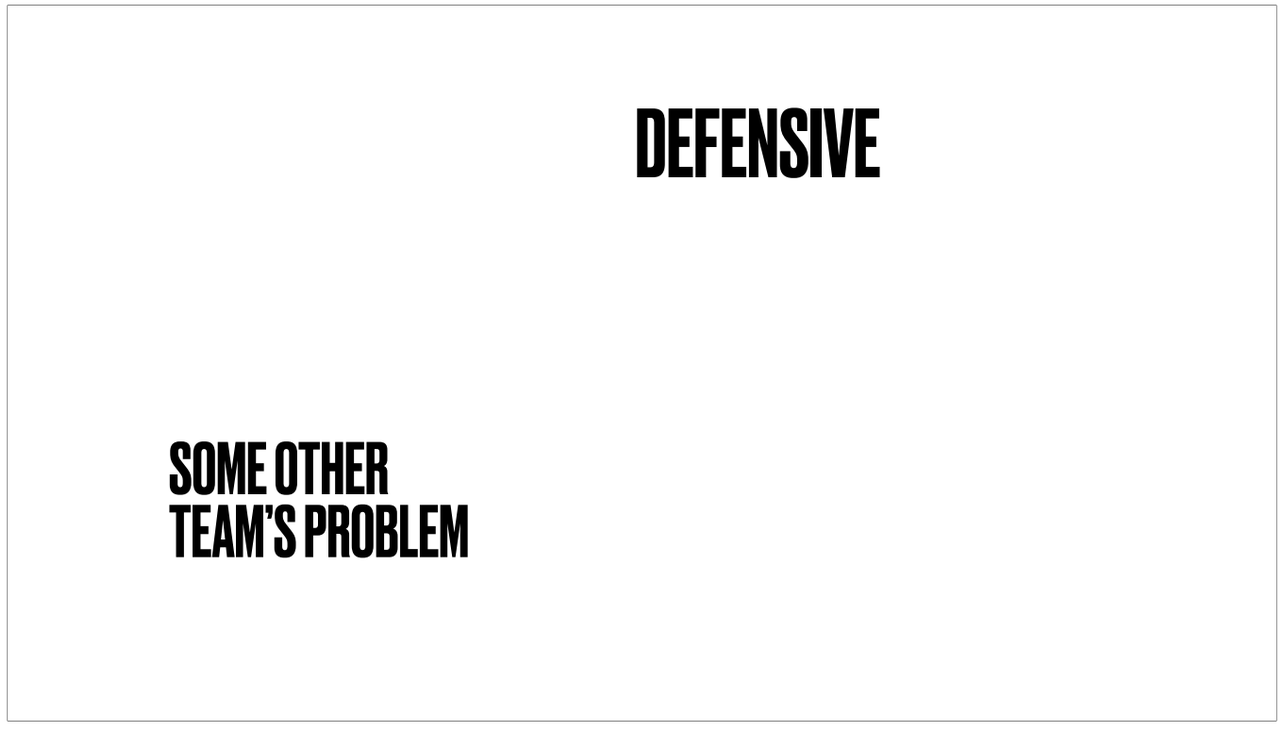
It's important to note that 271 devices have installed this via MSC since it was added in Q4 of 2024. Besides a bug that was reported during the initial rollout and this feedback, we've received no other reports of issues from Support, DevEx, or feedback on the Quickstart docs.

Issues with Documentation

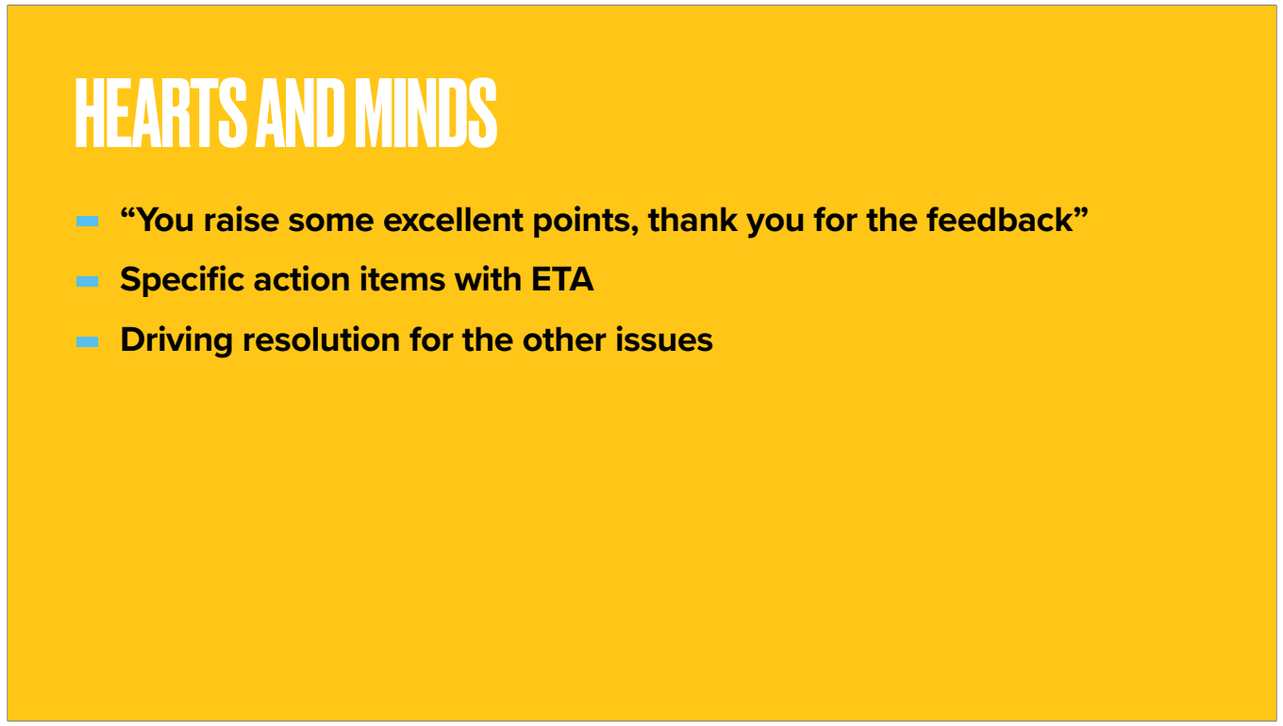
The user noted several issues with the getting started docs in general, mostly related to outdated or duplicated information.

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. 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.



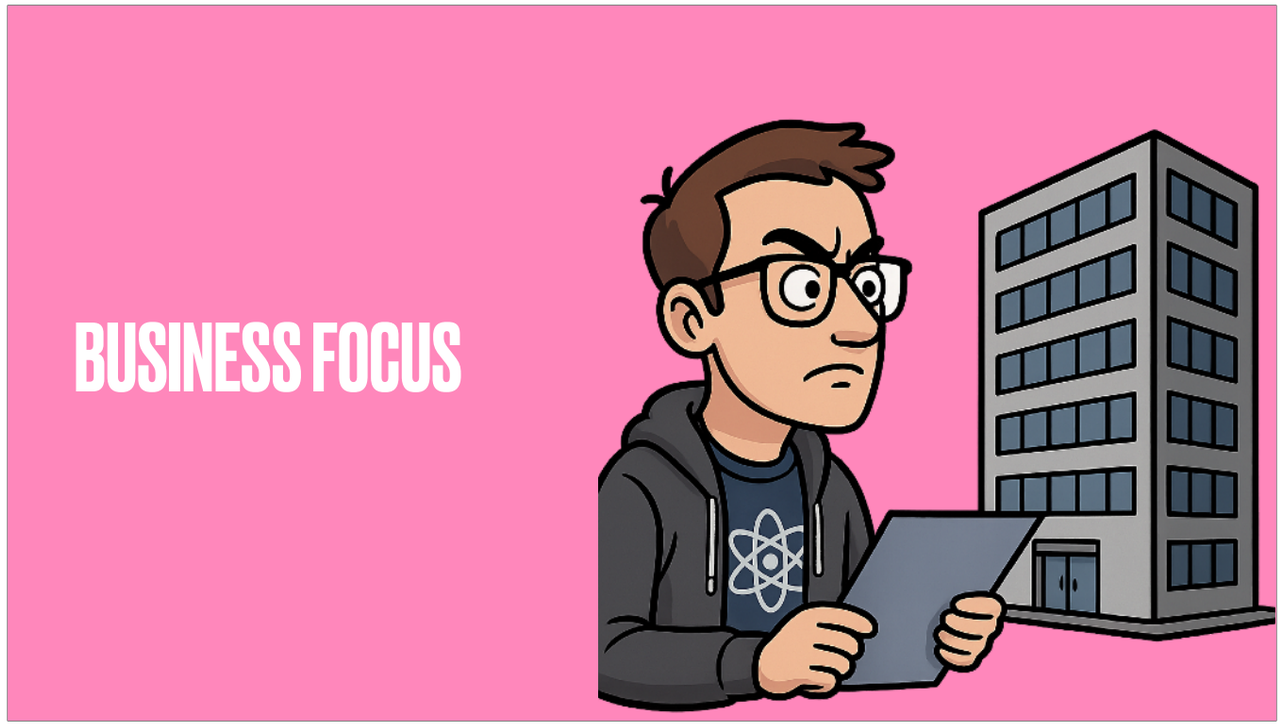
You're basically saying here we've fixed everything we can, the rest isn't our fault. So at best the exec is going to shrug their shoulders, not have a particularly good opinion of you or your team and move on. At worst, they're going to continue to insist that your solution is the wrong one, that your 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?



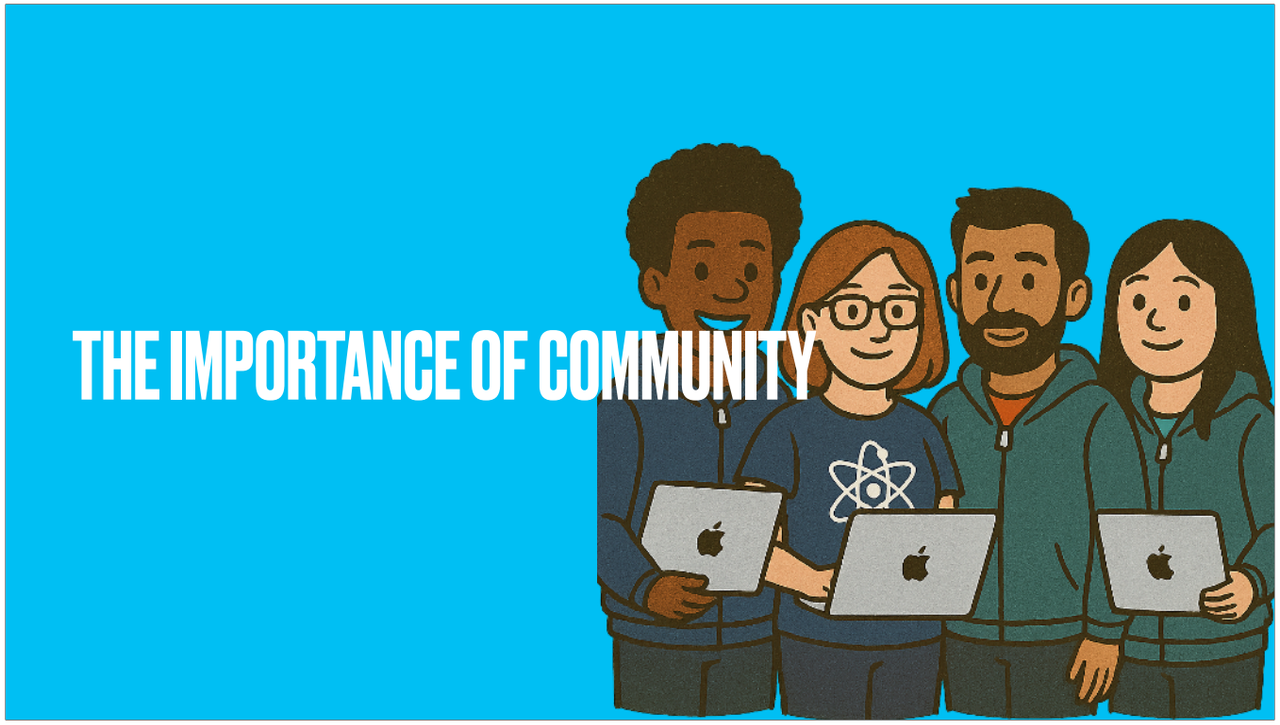
So it turns out that some of the items they raised have already been fixed in a later release. Rather than just saying that, you can acknowledge that they were absolutely right, highlight what you did to fix their issues, and if they have any other unresolved points, be specific about what the next steps 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.

So in our scenario before, how could you get those teams that owned the crappy docs to do what you asked them? Well obviously you have the weight of the VP behind you there. But they probably have a hundred other things to do that are also ultimately wanted by other very important people. How do you get them to work on what **you** want? It's not like you're their manager. You have no formal authority over them. I'm talking about influencing without authority.

Influencing without authority is as much of an art as it is a science. It all starts out with the points I've raised earlier. You've got recognized tech skills. You're able to tailor your message to it's audience, and you're able to find the opportunity in problems. You've been building relationships as well. You can't wait until you need something to reach out to people. You've built a wide network in advance — you know who to talk to on other teams, who owns which systems, and how to get things unstuck. You've been interacting with senior leaders, building your brand as someone they respect and trust to get things done. And by talking to those people, you know which levers you can pull, and how hard. Perhaps you know one manager won't ever get their team to do anything unless their director tells them to. Or perhaps another will only listen to a particular engineer on their team, so you need to seed the idea at the bottom first. For more on this, I strongly recommend reading this article. It's fantastic, and I refer people to it all the time.



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.



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.

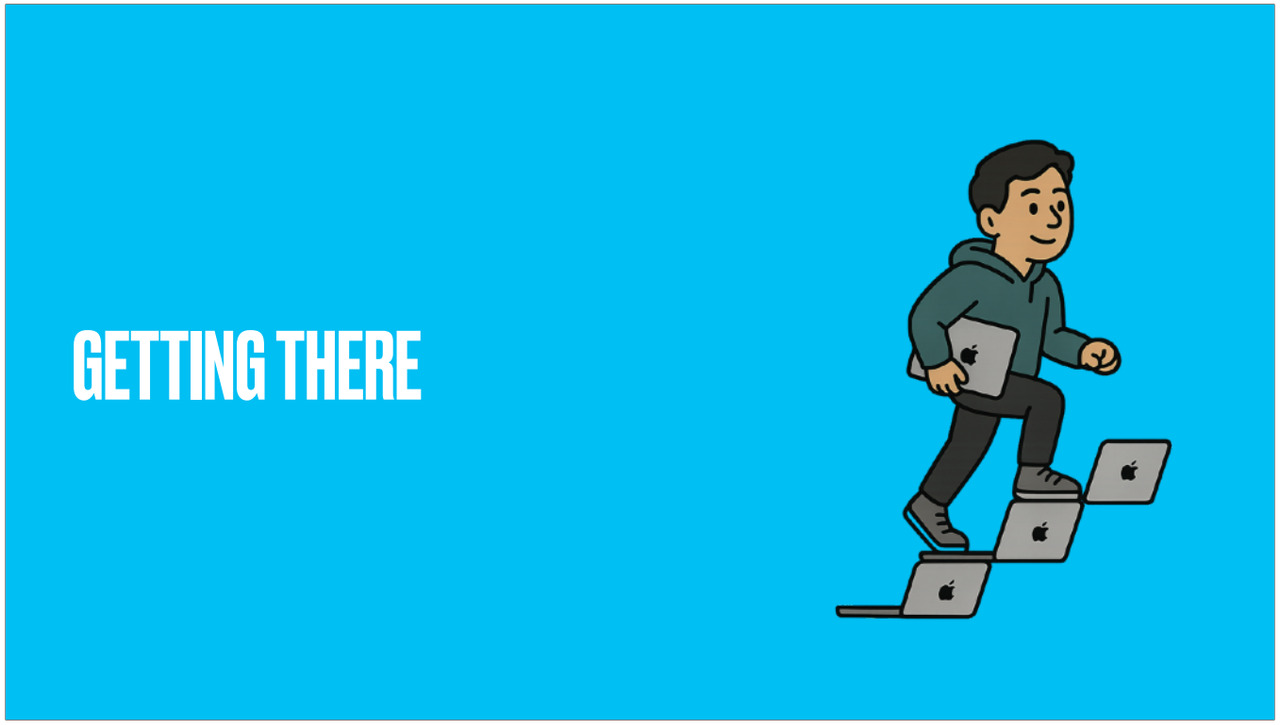
At Staff level, your reputation matters. Not just inside your company, but outside it too. Why? Because your visibility adds weight to your ideas. Because being part of a broader professional network gives you insight, inspiration, and allies. Because sometimes the best solutions come from people you've 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. Literally. Go to a manager, a director, someone adjacent to your org and ask:

"What problems are keeping you up at night?"
"What are you worried about over the next 6 to 12 months?"
"Where do you see risk or friction in the organization?"

Why do this? Because Staff Engineers don't just solve technical problems — they solve organizational ones. And the biggest, most strategic problems aren't always written down in Jira. Sometimes they're lurking in an exec's mental backlog. They might not even be fully formed yet — just vague concerns about scale, complexity, risk, or reliability. When you ask leaders what they're worried about, a few things happen:
- You get insight into the bigger picture.
- You build trust, because you're showing that you care about more than just your own backlog.
- And sometimes, you get handed a challenge that no one else has claimed — 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 sponsorship for promotion. Because when you solve a problem that matters to a leader, that leader becomes an advocate for you. So don't wait for perfect clarity or permission. Start with a conversation. Start with curiosity. Start by asking what keeps them up at night.

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.

So that’s it - we are hiring a security person with a focus on endpoint and a global head of IT support - if you are interested in learning more about either of those roles please chat with me.

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