First-Time Manager's Field Guide

Leading People Without Losing Yourself



Ryan Foster

First-Time Manager's Field Guide

Leading People Without Losing Yourself

by Ryan Foster

CONTENTS

I The Promotion Shock: What Changed and What Didn't

II The First 30 Days: Listen More Than You'll Want To

III Running 1:1s That Aren't a Waste of Everyone's Time

IV Giving Feedback That Actually Lands

V Receiving Feedback When You're the New Authority in the Room

VI Hiring: The Part Most Managers Fumble

VII Firing: When, How, and the Dignity Question

VIII The Performance Review, Actually Useful Version

IX Career Conversations: Not Just the Annual Check-In

X Managing Up: Your Boss Is Also Your User

XI Politics, Briefly (Because You Can't Pretend It Doesn't Exist)

XII When You're Stuck: Coaches, Mentors, and Books That Actually Help

XIII Burnout: Yours and Theirs

XIV The Kind of Manager You Want to Be Remembered As

CHAPTER I

The Promotion Shock: What Changed and What Didn't

The email arrived on a Tuesday. Subject line: "Congrats — you're official!" Your manager had CC'd HR, HR had CC'd someone in Finance, and by noon there was a Slack thread with seventeen emoji reactions and a gif of a person climbing a mountain. You were a manager now. Officially, formally, in-the-system manager.

Nobody sent you the job description.

That's not a metaphor. There was no document, no orientation module, no onboarding checklist that said: here is what you are now responsible for, here is how you will be evaluated, here is what you should stop doing immediately. There was a title change, a small salary bump if you were lucky, maybe a congratulatory lunch — and then Monday morning arrived and you were supposed to somehow know what to do with it.

Most people don't. Most people spend their first several weeks as managers doing a version of their old job while also trying to do their new one, which means they do both badly, feel guilty about both, and can't figure out why they're exhausted by Thursday afternoon despite the fact that nothing on their to-do list looks that hard.

This is the promotion shock. And it's almost universal.

Your Scoreboard Just Changed — Did Anyone Tell You?

Here's the thing about becoming an individual contributor at a company: the feedback loop is reasonably clear. You write the code, ship the feature, close the deal, design the slide deck — and you can look at the output and know whether it's good. The craft is visible. The result is yours.

Andy Grove, who ran Intel during one of the most consequential technology decades in history and who wrote High Output Management in 1983 with a clarity that still embarrasses most modern management books, put the shift bluntly: "A manager's output is the output of the organization under their supervision or influence." Not your output. Their output. The people reporting to you.

Read that again slowly, because it sounds obvious and it isn't. It means that from the day you became a manager, your personal productivity — the thing you'd been optimizing for the entire previous chapter of your career — became largely irrelevant as a performance metric. What matters now is whether your team ships, whether your team grows, whether your team solves the right problems without you being the one to solve them.

That's a complete inversion of how you got the job.

You became a manager, in most cases, because you were good at the individual work. You were the engineer who always found the elegant solution. You were the analyst who delivered the cleanest models. You were the salesperson who closed the accounts everyone else had written off. That visible, individual competence is what made someone tap you on the shoulder and say: you should lead a team. Which is a bit like watching someone win a marathon and saying: you should coach track. Related skills. Not the same skills.

Julie Zhuo, who started managing at Facebook at twenty-five — before she'd had time to develop much of an opinion about management theory — describes this realization in The Making of a Manager as one of the most disorienting parts of the early days. You spend years learning how to produce excellent individual work. Then the job changes and the measure of excellence is no longer something you can point to directly. It's ambient. It's downstream. A good manager's best work often looks, from the outside, like nothing in particular.

That's uncomfortable for people who got where they are by being visibly excellent.

Ryan's First Week: A Comedy of Misdirected Competence

I want to tell you about the first week. Mine specifically.

I became a manager on a Monday in October. My team was six engineers, and my old role — senior engineer, the job I'd held for two and a half years — was going to be backfilled by someone already on the team who I'll call Marcus. Marcus was good. Excellent, actually. The transition made sense.

By Wednesday of that first week, I had answered fourteen Slack messages that were meant for Marcus.

Some of them were architectural questions about the service I'd spent eighteen months building. Some were pull request reviews I had technically excellent opinions about. One was a customer escalation that I jumped on because I knew the codebase and I could fix it faster than anyone else on the team. I fixed it in forty minutes. Felt great.

Meanwhile, I had not done any of the things that were my actual job. I had not scheduled 1:1s with any of my six reports. I had not talked to my manager about what the next quarter's priorities were. I had not read the two strategy documents sitting in my inbox. I had not thought for five minutes about who on my team was frustrated, who was growing, who was about to quit, or who needed something from me that had nothing to do with the pull request queue.

I was a manager who was very efficiently doing an engineer's job.

My manager — who had been patient about this for exactly four days — finally sent me a message that said, roughly: "You're not an engineer anymore. Stop answering those." It was blunt in the way that the best management feedback is blunt. Not mean. Just accurate. And it stung precisely because it was true.

What I was doing wasn't laziness or incompetence. It was comfort. The engineering work was familiar. I knew how to evaluate it. I knew when I was doing it well. The management work — the 1:1s, the strategy alignment, the attention to team health — was murky and new and I had no idea how to tell whether I was doing it well or not. So I kept gravitating back to the thing I knew.

This is, it turns out, one of the most common failure modes of new managers. Camille Fournier calls it "staying in the weeds" in The Manager's Path, and she's appropriately unsentimental about it: the inclination to keep doing the technical work isn't just a time management problem, it's a signal that you haven't fully internalized what the job now requires. It's not that the old skills become worthless. It's that they can become a hiding place.

What Actually Changed

Let's be specific, because vague advice about "letting go of individual work" is not going to help you on a Wednesday afternoon when you have an opinion about someone's database schema and nobody's stopping you from typing it into Slack.

Your success metric changed. This is Grove's point, and it's the most important one. You are now evaluated, whether or not anyone has made this explicit to you, on the collective output of your team. If your team ships good work, you are doing well. If your team is disorganized, underthrowing, or quietly miserable, you are not doing well — even if you personally are closing tickets at a legendary pace. The scoreboard changed. Your job is to internalize that and let it guide where you put your attention.

Your relationship with time changed. Individual contributors, even senior ones, can usually carve out two or three hour blocks of uninterrupted focus. Flow state is a real thing and it's productive. Manager time doesn't work that way. Your time is, in a fundamental sense, other people's time. Your team needs access to you. Your own manager needs updates from you. Stakeholders need decisions from you. You will have fewer long, uninterrupted stretches, and fighting this fact will make you miserable. The better adaptation is learning to do your real cognitive work — the thinking, the prioritizing, the relationship-tending — in smaller, more distributed windows. It's a different rhythm. It takes months to get comfortable with.

Your relationship with information changed. As an individual contributor, you accumulated expertise over time and that expertise was your value. As a manager, you're often the least technically specialized person in the room, and that's fine — what you now need is different information: what is the team worried about, where are the blockers, what's the morale like, who is about to hit a wall. You need to build new information-gathering instincts. The old ones — staying close to the codebase, knowing the product cold, reading every spec — are now partially someone else's job.

Your identity at work changed. This one doesn't get talked about enough. A lot of people derive significant satisfaction, maybe most of their professional satisfaction, from being good at a craft. The engineer who can debug anything. The writer who always finds the right sentence. The researcher who knows the literature. That identity doesn't disappear when you become a manager, but it gets complicated. Because now you are

responsible for a team full of people who are better at those crafts than you're going to be at your new stage — and your job is to help them be excellent, not to be the most excellent person yourself. For people whose professional self-worth is built on craft, this transition can be genuinely destabilizing. It's worth knowing that going in.

What Didn't Change

Before you close this book and ask for your old job back: the flip side matters.

Management is real work. This should be obvious, but the way some people talk about it — the administrative overhead, the meeting-heavy calendar, the HR paperwork — can make it sound like something you do instead of working. It isn't. It's a different kind of working, and done well, it's cognitively demanding in ways that sneak up on you.

Thinking is still your job. The quality of your decisions matters enormously. Which problems does the team prioritize? How do you structure the team to minimize coordination costs? When do you escalate versus absorb? These aren't questions you answer by attending more meetings. They require thinking, and managers who don't protect time for thinking end up reactive in a way that compounds every other problem their team faces. Grove was adamant about this: managers have to do the intellectual work of understanding their organization, which means reading, observing, talking to people outside your immediate team, and sitting with problems long enough to actually understand them.

Craft still matters — the craft just shifted. You used to care about code quality or design elegance or analytical rigor. Now you care about communication quality, meeting structure, feedback clarity, and team ritual. These are learnable skills. The ones you already have — clear thinking, attention to detail, the ability to recognize when something isn't quite right — transfer. They apply to a new domain. The engineer who couldn't stand sloppy code often makes an excellent manager of engineers, because that same precision gets redirected toward clarity of expectation and quality of process.

Relationships still drive everything. If you were good at building trust with colleagues as an individual contributor, you have a head start. The dynamic with your direct reports is different — there's a power differential now that didn't exist when you were peers — but the underlying mechanics of trust are the same ones you've always known: do what you say, be honest, show up when it counts.

You still have to do boring administrative work. It is not glamorous. Neither was the TPS-report equivalent in your old job. You will still have to fill out things, approve things, attend things, and sign off on things that feel like they were designed specifically to frustrate you. This did not change. Welcome to organizations.

The Peer-to-Boss Transition: The Specific Weird Thing Nobody Warns You About

If you were promoted from within your team — if the people you now manage are the same people you ate lunch with last month, the same people you complained about your old manager to — you have a specific and delightful challenge ahead of you.

The relationship has changed even though nothing visible has changed. You are now their manager. That means you hold information about their compensation, their performance, their future at the company. It means that when they tell you something in frustration, you have a different kind of responsibility for it than you did before. It means that the casual solidarity of being fellow ICs — "management is clueless," "this process is broken," "can you believe that decision?" — is now something you have to step back from, because you are, to some degree, management.

This doesn't mean you become a robot. It means the register of the relationship shifts. Some of your former peers will adapt to this quickly and naturally. Some will test the boundary for months. A few will resent you regardless of how well you handle it, and this is genuinely painful the first time it happens.

Zhuo writes honestly about the first time she had to give critical feedback to a former peer — someone she'd been close with — and the way it strained the friendship. Her observation: the friendship was real, but it was a friendship between peers. The new relationship isn't less real; it's just different. Trying to maintain the old dynamic while exercising the responsibilities of the new one is the thing that actually damages trust, because it creates a fundamental confusion about what the relationship is.

The cleanest approach, uncomfortable as it is, is to name the change directly with your team, early. Something like: "This is a different relationship than what we had, and I want to be straightforward about that. My job now is to support your work and your growth. I take that seriously and I'm still figuring out what it looks like in practice — and I want us to figure it out together." It's not a speech. It's just an acknowledgment that the thing everyone is awkwardly tiptoeing around is real.

The Output You Can't See

Here is something nobody tells you and that takes most managers a year or two to fully absorb: a lot of your best work will be invisible.

The engineer who would have quit if you hadn't caught the signals and had one real conversation with her — she didn't quit, so there's no visible intervention to point to. The project that ran smoothly because you did the pre-work of aligning two stakeholders who would otherwise have been fighting over scope — there's no artifact of that alignment. The team member who got the stretch assignment because you made the case for him in a calibration meeting you never told him about — he'll think he earned it entirely on merit, which is exactly how it should feel.

This is Grove's point made visceral. Your output is the organization's output. The best you can say, in your quieter moments, is that things went well and you played some part in that. The precise part is hard to attribute. The feedback loop is long and diffuse in a way that individual contributor work usually isn't.

There's a corollary to this that's worth sitting with: it means that people who need immediate, visible recognition for their work often struggle in management roles. Not because they're bad people or even bad managers, but because the reward structure of the job doesn't deliver that. If you need to see the thing you built and know it's yours, management is going to be a grinding adjustment. Some people get there by shifting what they find meaningful. Some find, after honest reflection, that they'd rather stay excellent individual contributors — which is a completely legitimate choice and one more companies should make easier to make.

What Your First Month Actually Looks Like

Stop optimizing for the old metrics. That's the first thing.

This doesn't mean disengage from the technical work entirely. There's a version of "you're not an engineer anymore" advice that tips too far and leaves new managers uselessly detached from the work their teams are doing. You need enough context to understand what your team is building, where the hard problems are, and when something is going wrong. What you're weaning yourself off is being the one who solves the problems, not being the one who understands them.

Here's a rough frame for the first four weeks:

Week one is observation. You have opinions. Keep most of them to yourself. You don't know yet what you don't know, and the surest way to reveal this is to start making changes before you understand the terrain. (More on this in the next chapter.) Watch how the team works. Sit in meetings. Read the documents that exist. Notice who talks to whom and who doesn't.

Week two is conversation. Start the 1:1s. Not structured, agenda-heavy 1:1s — those come later. Just conversations. Ask how people are doing. Ask what they're working on. Ask what's getting in their way. Don't fix anything yet. Just listen at a rate that will probably feel unnatural if you're used to being an action-oriented individual contributor.

Week three is alignment upward. Have a real conversation with your manager about what success looks like in the next ninety days. Not in a career-anxiety way — in a genuine "I want to make sure I'm focused on the right things" way. You may be surprised by what they say. You may discover that their priorities for your team are somewhat different from what you assumed. Better to find out in week three than week thirteen.

Week four is where you start, carefully, to add your perspective. Now you have some data. You've had the conversations. You've watched the team work. You have hypotheses about what's going well and what isn't. You can start to share those — first with your manager, then with the team — as questions more than declarations. "I've noticed that our planning process seems to create a lot of rework in the first week of each sprint — does that match your experience?" is a better opening than "I want to change how we do planning."

This isn't timidity. It's basic epistemology. You don't know enough yet. Admitting that isn't weakness — it's the only accurate representation of your situation.

The Manager You Are Not Yet

One more thing, and then we get to the practical work.

There is a version of you, somewhere in the future, who is good at this. Not perfect. Not the manager in the Harvard Business Review case study. But genuinely good — someone who gives feedback that actually helps, who runs team meetings that are worth attending, who makes hiring decisions they mostly don't regret, who can have a hard conversation on a Tuesday and not lose sleep about it until Friday.

That person does not exist yet. You are not them yet. This is fine. It's where every manager starts, including the ones who are now excellent, including the ones who wrote the books we'll keep referencing in these pages.

The distance between where you are and where you're going is not traveled by reading the right things or attending the right workshops, though both help at the margins. It's traveled by doing the work, badly at first, getting feedback, adjusting, and doing it again. Andy Grove ran Intel. Camille Fournier has CTO'd multiple companies. Julie Zhuo built product teams at Facebook for a decade. All of them have pages in their books that are essentially: "I got this wrong and here's what I learned." The learning is the path. There isn't a shortcut around the bad calls.

What you can do, right now, in the next forty-eight hours, is start making the shift explicit to yourself. Not just "I am a manager now" as a title and a fact — but "my job has changed in specific ways, and I'm going to behave accordingly." Answer fewer Slack messages that belong to someone else. Block one hour this week with no agenda except thinking about your team. Schedule the first 1:1 with the person on your team you know least well.

The scoreboard changed. You can play by the old rules for a while — most people won't stop you — but it'll catch up with you. And more importantly, it'll catch up with your team, who didn't ask to be managed by someone who's still solving yesterday's problems.

They asked to be led. That starts now.

CHAPTER II

The First 30 Days: Listen More Than You'll Want To

The day you get promoted, a small but powerful voice activates somewhere in your chest. It tells you that you need to do something. Make a mark. Justify the title. Show everyone who gave you this role that they were right.

That voice will get you into trouble faster than almost anything else in your first month.

The Reorg That Almost Wasn't

Nine days into his first management role, Ryan had a plan. A good one, he thought. His team of seven engineers was organized in a way that made sense when the product had three features — now it had eleven, two of the features were scaling hard, and the org structure hadn't caught up. He'd noticed it within forty-eight hours. He sketched it out on a Sunday evening: two small pods, each with a clear product surface, each with a senior engineer leading the technical direction. Clean. Obvious. Efficient.

He floated the idea in a casual conversation with two of his reports on day nine.

The conversation stopped. Not paused — stopped. One of them, a senior engineer named Marcus, said something noncommittal and looked at the floor. The other one, Priya, said she had a meeting to get to and left.

Ryan found out three weeks later, from a peer he'd finally gotten comfortable enough to confide in, what he hadn't known: Marcus and Priya had been up for the same senior engineer promotion two years earlier. Marcus had gotten it. The relationship had never fully recovered. The reorg Ryan was proposing would have put Priya in a pod technically adjacent to Marcus's. Not reporting to him — but in his orbit. That was enough.

He'd spent nine days understanding the code. He'd spent zero days understanding the people.

Why We Do This

There's a specific anxiety that comes with being new to a management role, and it's different from the anxiety of being new to any other job. When you joined as an individual contributor, you had roughly ninety days before anyone expected you to be fully productive. There was scaffolding. There was onboarding. There was implicit social permission to not know things yet.

As a new manager, you often get none of that. You're expected to be leading on day one. Team members start bringing you problems immediately, sometimes the morning of your first day, because there's been a queue

building up for weeks that your predecessor couldn't clear or your role was vacant and nobody had been absorbing the managerial work. The implicit message is: you're in charge now. Act like it.

So you do. You act like it. Which means making decisions, proposing changes, showing decisiveness. All the things you think management looks like from the outside.

The problem is that you're making decisions with about two percent of the information you actually need. You know the surface of the terrain. You don't know the geology.

Michael Watkins, who spent years studying leadership transitions at Harvard Business School, documented this pattern in enough detail to name it: new leaders, he found, consistently overestimate how much they know about an organization and underestimate how much that organization knows about itself. The team you just inherited has been accumulating context for months or years — about what's been tried, what failed, what the unofficial rules are, who the real decision-makers are. You have none of that. None.

Acting before you have it isn't decisive. It's expensive.

The 30-Day Sprint

What follows is a concrete framework for your first thirty days. It's not thirty days of doing nothing — that's its own mistake. It's thirty days of directed listening, which is harder than it sounds and produces more useful output than almost anything else you could do.

The frame comes partly from Julie Zhuo's The Making of a Manager, where she's explicit that a new manager's first job is to understand the team and the work before reshaping either. "Your job," she writes, "isn't to have all the answers. It's to create the conditions where the best answers can emerge." That's easier to believe when you're not sitting in your first team meeting with everyone looking at you expectantly.

It has three phases: stakeholder interviews, the team State of the Union, and your personal README. By day thirty, you'll have a map. A real one.

Phase One: Stakeholder Interviews (Days 1–14)

Start with the people who have opinions about your team — because they all do. Everyone above you, beside you, and in adjacent functions has a view about what your team does well, what it does badly, and what they wish it would do differently. Most of them have never been asked.

Ask them.

Schedule thirty-minute conversations with eight to twelve people. Your own manager, obviously. Two or three of your manager's peers who interface with your team regularly. Key stakeholders in product, design, sales, or whatever functions your team depends on or serves. If your team has internal customers, talk to them.

Use the same five questions with everyone. Consistency matters here — you're collecting qualitative data, and variance in the questions creates noise you don't need.

The five questions:

- What does our team do really well right now?
- What does our team do that frustrates you?
- If you could change one thing about how we work together, what would it be?
- What's the most important problem you think the team hasn't solved yet?
- What do I need to understand about this organization that won't be obvious from the outside?

That last question is the one that gets you the real answers. It signals that you know you don't know things, which is both honest and — counterintuitively — confidence-building. People trust leaders who know their own limits more than leaders who pretend they don't have any.

Take notes. Not on a laptop if you can avoid it — a notebook signals that you're present, not processing. After each conversation, spend ten minutes writing a summary before the details fade. What surprised you? What came up that you'd heard before? What seemed emotionally charged in a way that warrants follow-up?

A few things will happen across these conversations. Some themes will repeat — you'll hear the same frustrations from three different people, which means they're real and not just individual gripes. Some opinions will contradict each other violently, which tells you something about competing priorities or old organizational conflicts you'd have walked straight into. And some things will only come up once, from one person, in an offhand comment that turns out to be the most important thing anyone tells you.

You will not know which category anything falls into until you've done all the conversations. That's the point.

Your Direct Reports Aren't Stakeholders — They're the Subject

Run separate conversations with your direct reports. This isn't the same as 1:1s — Chapter Three covers those in depth. These are onboarding conversations. One-time, structured, explicitly framed as "I'm here to understand your experience before I do anything."

The questions shift:

- What's going well for you on this team right now?
- What's getting in your way — things only a manager could remove?
- What do you wish the previous manager had done differently?
- What are you working on that you're most excited about?
- What do I need to know about how to work with you well?

That last question does something useful: it puts the direct report in the expert seat. They know things about themselves that you don't, and most people are rarely asked to share that information so directly. You'll get answers that range from "I prefer written feedback over verbal" to "I'm dealing with a family situation that

affects my availability on some mornings" to "I need a lot of autonomy or I disengage." All of it is useful. None of it should be filed away and forgotten.

Camille Fournier makes the point in The Manager's Path that new managers often inherit team dynamics they didn't create and don't initially understand — and that treating those dynamics as problems to fix before you understand their origin is how you create new, worse problems. Your job in these early conversations isn't to assess. It's to listen without flinching. If someone tells you something uncomfortable about how the team has been run, thank them for telling you and keep asking questions.

Do not share your opinions about what you're hearing. Not yet. Not with your reports, not with your peers, not in passing comments in the hallway. The information flows one direction for now: in.

Phase Two: The State of the Union (Days 15–21)

Around day fifteen, bring the whole team together for what you'll call — internally, in your own planning — the State of the Union conversation. You can call it whatever you want externally. A team kickoff, a reset, a roundtable. The name matters less than what happens in it.

The purpose is to give the team a chance to speak collectively about the work. This is different from individual 1:1 conversations — group conversations surface different things. Group norms, shared frustrations, collective pride. Things that feel too small to raise individually but are big enough that everyone feels them.

Run it with a light structure. Four questions, roughly twenty minutes each:

- What's working? Ask people to name specific things — projects, processes, team habits — that they think are genuinely good and shouldn't be touched.
- What's broken? Not as a complaint session, but as honest diagnosis. What slows the team down? What creates unnecessary friction?
- What's unclear? Priorities, roles, expectations. Where does the team feel like it's guessing?
- What does the team want to be known for in a year? This one shifts the energy from diagnosis to aspiration. It's also diagnostic in its own way — you'll learn a lot about the team's identity and ambitions from how they answer it.

Facilitate, don't participate. This is not a meeting where you share your own views. Take notes on the whiteboard or in a shared doc so everyone can see what's being captured. At the end, read back the themes you heard and ask if you missed anything.

Then do something that first-time managers almost never do: close the loop. Send a written summary within forty-eight hours. "Here's what I heard. Here's what surprised me. Here's what I'm going to spend more time understanding before I make any decisions." This does two things. It shows the team you actually listened. And it creates accountability — you've told them what you heard, so they'll notice if you proceed as if you didn't.

Phase Three: Your Personal README (Days 10–25)

Concurrent with the listening work, write your personal README.

Julie Zhuo describes this exercise in The Making of a Manager as one of the most useful things a new manager can do. The idea is simple: write a short document — two pages, three at the outside — that tells your team how you operate. Not your vision for the team, not your goals for the quarter. How you work. What you value. What frustrates you. How you like to communicate. What you're bad at. What you need from the people you work with.

This is uncomfortable to write for two reasons. First, it requires self-knowledge, which takes effort. Second, it requires admitting your limitations to the people you're supposed to be leading, which feels risky.

It isn't. It's the opposite of risky.

When you're explicit about how you work, two things happen. Your team stops guessing — and they will guess, often incorrectly, which creates misalignments that are hard to diagnose. And you signal that you're a person, not a function. People work harder and more honestly for managers they experience as human.

A useful README covers six things:

What I value most. Be specific. "I value transparency" is too generic to be useful. "I value knowing about problems before they become emergencies, even if you're not sure it's your problem to raise" is actionable.

How I communicate best. Do you prefer Slack for quick questions and email for anything that requires a decision? Do you need an agenda before a meeting or do you prefer to think live? Say so.

What I'm still learning. You're new to management. You're going to make mistakes. Naming a few things you know you're not yet good at — not fishing for reassurance, just stating facts — creates space for your team to give you feedback without feeling like they're attacking you.

What I need from you. Direct communication, a heads-up on blockers, a challenge when I'm wrong. Whatever it actually is for you.

What will frustrate me. Surprises, blame without accountability, problems that waited three weeks to surface. You don't have to say this punitively. You're just telling the truth.

What I promise you. What your team can actually count on from you as a manager. Clear feedback. Context when direction changes. Credit when things go right.

Share it in your team channel. Invite questions. Be willing to revise it. It's not a contract — it's a conversation starter. The act of sharing it, more than the document itself, is what builds the foundation you need.

What Day Thirty Actually Looks Like

By day thirty, you should have: notes from ten or twelve stakeholder conversations, notes from every direct-report onboarding conversation, a synthesis from the State of the Union, and a draft README you've already shared.

You should also have a map. Not a perfect map. Not a map that tells you everything. But a map that shows you where the terrain is stable and where it's uncertain, which problems are real and which are symptoms of something deeper, which relationships are warm and which are carrying old tension.

Andy Grove wrote in High Output Management that information is one of the manager's most critical inputs — that a significant part of the job is simply collecting it, from conversations, from observation, from written reports, from the gaps between what people say and what they do. He was describing experienced managers doing ongoing work. The principle is even more true for new managers in their first month, when almost everything they think they know about the situation is provisional.

Your job in week four is synthesis, not action. Take everything you've collected and ask three questions:

What are the three things the team most needs right now? Not what you want to give them. What they need — the blockers, the gaps, the festering problems that nobody has cleared.

What's working that you should actively protect? New managers break things accidentally by not realizing those things exist. Name them explicitly so you don't accidentally step on them while pursuing something else.

Where are you still blind? What did you not hear enough about? Whose perspective is missing? Who didn't tell you the real thing?

Then write a brief document — for yourself, not for anyone else — that answers those questions. Date it. You'll want to look at it in six months.

What You're Not Doing in Thirty Days

This matters as much as what you are doing.

You're not reorganizing the team. Not proposing new processes. Not changing the sprint cadence, the meeting structure, the on-call rotation, or the engineering standards. Not unless something is actively on fire — and "actively on fire" means a real emergency, not a suboptimal situation.

You're not rendering verdicts on people. In thirty days, you have nowhere near enough information to assess whether someone is performing well or badly, whether they're a culture fit or not, whether they belong on the team. You have impressions. Impressions are not assessments.

You're not making your predecessor's choices your problem to fix right now. Every team you'll ever inherit was shaped by the manager before you. Some of those choices were good. Some were bad. Some were compromises between bad options. You can't know which is which yet, and relitigating decisions you weren't there for is a fast path to undermining the team's confidence in its own history.

Fournier is blunt about this in The Manager's Path: "The worst thing you can do as a new manager is come in and immediately start changing things just to make your mark." Not because change is bad. Because change without understanding is expensive, often irreversible, and signals to the people who've been there longer than you that their experience doesn't count for much.

You will feel enormous pressure to act before you're ready. Some of it will be internal — the anxious voice we talked about at the beginning. Some of it will be external, from stakeholders who are excited about the new management energy and want you to solve their pet problems immediately. Learn to say, in a way that's warm but firm: "I'm still building my understanding of the team's priorities. I want to make sure I get this right rather than fast. Can we revisit in a few weeks?"

Most people will respect this. The ones who don't are usually the ones with a personal stake in you acting before you understand what you're doing, which is its own useful data point.

The Thing About Trust

Here's what thirty days of listening actually builds, underneath the tactical value of all the information you're collecting.

Trust.

When people feel heard — actually heard, not just accommodated — they extend good faith. They give you the benefit of the doubt when you make a decision they don't fully understand. They tell you things sooner, when there's still time to act on them. They invest in your success because you've demonstrated that you're invested in understanding them.

This matters because you will make mistakes. Everyone does. Your first few months will include at least one decision that turns out wrong, one conversation that lands badly, one situation you handle more clumsily than you'd like. The team's tolerance for those mistakes — the speed with which they assume good intent and move past them — is largely a function of the trust you built before the mistakes happened.

Ryan's day-nine reorg idea didn't blow up because he shared it. It didn't blow up at all, ultimately — he shelved it, spent three more weeks listening, and eventually proposed a different structure that Marcus and Priya both signed off on. But it was close. And the reason it didn't blow up wasn't that he'd been lucky. It was that the two people he'd floated it to — Marcus and Priya both — had experienced enough of his listening in those nine days to give him a chance to pull back rather than going around him to complain.

Goodwill is a bank account. You can only spend what you've deposited.

Spend the first thirty days making deposits.

The impulse to act quickly is understandable. It comes from a real place — the desire to be useful, to justify the role, to show the people who believed in you that they were right. None of that is wrong. But the most

useful thing you can do in your first month is build the foundation that makes everything after it faster, cleaner, and less likely to blow up in your face.

Listen. Map the terrain. Write down what you learn.

The plans can wait three more weeks. The people can't.

CHAPTER III

Running 1:1s That Aren't a Waste of Everyone's Time

Picture this: it's 10 a.m. on a Tuesday. You and one of your direct reports are sitting across from each other — or staring at the same video-call grid — and she's telling you that the API migration is at 73%, the design review is scheduled for Thursday, and the staging environment went down yesterday but came back up in twenty minutes. You nod. She nods. You ask if there's anything else. She says no. You both go back to your desks and cross "1:1 with Sarah" off the calendar, vaguely relieved it's done.

That was not a 1:1. That was a status report with two people in the room.

The distinction matters because the status report costs you both thirty minutes and produces information you could have read in a Slack message. The real 1:1 — the kind Camille Fournier describes in The Manager's Path as one of the most important tools in a manager's toolkit — is something different entirely. It's where you find out that Sarah is quietly applying elsewhere because she feels invisible on the team. It's where you learn, six weeks before it becomes a crisis, that the relationship between your two senior engineers has curdled into something that is slowing down every code review. It's where a person who would never walk into your office to say "I'm struggling" will, given the right container and the right question, actually tell you the truth.

Most first-time managers run 1:1s as status reports. This chapter is about stopping that.

Why the 1:1 Is Your Highest-Leverage Hour

Andy Grove was not a warm and fuzzy guy. He ran Intel with a kind of relentless precision that produced some of the most influential management writing of the last half century. So when Grove wrote in High Output Management that the one-on-one meeting is "the principal way the supervisor can maintain a working relationship with the subordinate," and that its purpose is "mutual learning and the exchange of information," he wasn't being sentimental. He was doing math.

Think about the leverage. You have, let's say, eight direct reports. One well-run weekly 1:1 with each of them gives you roughly 400 hours per year of structured signal about what's actually happening on your team. That's 400 hours of early warning. 400 hours of course correction before small drift becomes large damage. 400 hours where you can find out what's blocking someone who is too professional to complain publicly, or what's energizing the person you've been underestimating.

Managed badly, that same 400 hours is overhead. Pure friction. The kind of recurring calendar event that people start declining with thin excuses.

The difference between those two outcomes is almost entirely about who's running the meeting and what questions get asked.

It's Their Meeting. Seriously.

Here is the thing most new managers don't hear clearly enough: the 1:1 belongs to your direct report. Not to you.

Julie Zhuo, writing about her years managing design teams at Facebook in The Making of a Manager, is unambiguous about this. The 1:1 is the employee's opportunity — to share what's on their mind, to raise problems, to talk about their work, their growth, their frustrations. The manager's job is to create conditions in which that sharing can happen. That is not the same as running the meeting.

What this looks like in practice: your direct report sets the agenda, or at least leads with it. You ask questions more than you answer them. If you catch yourself talking for more than two consecutive minutes unprompted, you've probably flipped back into broadcasting mode.

This is harder than it sounds. New managers are often promoted partly because they're good at having answers, and the instinct to provide answers doesn't evaporate the moment someone hands you a management title. You sit down with someone who mentions a problem, and every fiber of your problem-solving brain fires. You want to help. You want to demonstrate competence. You want to say the useful thing.

Resist it. The most useful thing you can do in most 1:1s is ask a better question.

The practical test is simple: who does most of the talking? In a well-run 1:1, the direct report speaks somewhere between 60 and 75 percent of the time. If you're talking more than that, the ratio is inverted and you're probably leaving the meeting with less information than you walked in with.

A Concrete Agenda (That Doesn't Feel Like a Form)

Structure matters, even when the conversation feels loose. Without some scaffolding, 1:1s drift — toward whatever is most recent, most urgent, or most comfortable to discuss, which is rarely the thing most worth discussing.

Here's the template that works, adapted from Fournier's framework in The Manager's Path:

1. Their agenda first (10–15 minutes). Send a shared doc or a Slack thread before the meeting with a standing prompt: "What do you want to cover today?" Let them add to it before you walk in. This does two things. It signals that the meeting is theirs to shape, and it forces some reflection — people who actually think about what they want to discuss before the meeting tend to surface better issues than people who show up cold.

2. A check-in question (2–3 minutes). Not "how are you?" — that question has a socially mandated answer of "fine." Something more specific. "What's been the best part of this week so far?" or "What's felt most draining lately?" The answer is often casual, but occasionally it cracks something open. You're calibrating

energy and affect. You're listening for signals that don't appear in ticket trackers.

3. A focused topic (10–15 minutes). Not every 1:1 needs one, but most should have at least one substantive topic you go into depth on. A problem they're stuck on. A relationship that needs untangling. A skill gap they're trying to close. This is the coaching window — and it requires you to stay in question mode. More on this below.

4. Your items (last, and brief). Yes, you have things to communicate. Announcements, context from above, process changes. Put them last and keep them short. The meeting ending with "okay, and just from my side — the Q3 planning timeline shifted, I'll drop the updated dates in Slack" is fine. The meeting starting with fifteen minutes of your agenda items is not.

5. A close. "Is there anything you wanted to cover that we didn't get to?" Said sincerely, it functions as a safety valve. Sometimes the most important thing is the thing someone was nervous to raise.

Total time: 30 minutes works for most relationships and most weeks. 45 if things are complex or the relationship is newer. Grove recommended at least an hour for subordinates in technical roles where the work is deep — and there's a case for that cadence quarterly even if your weeklies run shorter.

The Questions That Actually Surface Problems

Standard 1:1 questions produce standard 1:1 answers. "How's the project going?" produces "good, just working through some stuff." That's not useful information.

The questions that produce real signal are more specific, slightly more personal, and marginally more uncomfortable to ask. Here are eight that work:

"What's blocking you right now that you haven't mentioned yet?" The "haven't mentioned yet" does real work. It acknowledges that some things don't make it to the surface naturally and gives explicit permission to raise them.

"If you could change one thing about how the team operates, what would it be?" Useful for surfacing process friction, interpersonal static, and organizational dysfunction that people assume is just immovable background noise.

"What part of your work feels least connected to the things you care about?" This is a career question disguised as a workload question. The answer tells you a lot about engagement and about whether someone is quietly considering exits.

"Who on the team have you found it hard to work with lately, and what's making it hard?" This question requires some trust in the relationship before it will yield an honest answer. But once it does, you learn things that simply don't appear in any other channel.

"What do you wish I understood better about your work?" Simple, and almost always productive. People have things they want their manager to know. This gives them a clean lane.

"What's one thing I'm doing that's getting in your way?" This one takes courage to ask, and you have to actually be ready to hear the answer without deflecting. More on receiving feedback in chapter five. But the manager who asks this question regularly, and does something visible with the answers, builds a level of trust that compounds.

"What would make this year feel like a success for you, personally?" Long-range, a little vulnerable. Ask it every quarter and the answers shift in interesting ways.

"Is there anything you've been wanting to bring up but haven't found the right moment for?" Close cousin to the safety-valve question at the end. Best asked midway through, not as a formality at the end, when there's still room to actually go somewhere with the answer.

You don't ask all of these in one 1:1. You rotate. You read the room. But you build a habit of asking something that goes one layer deeper than the project surface, every week, until the honest answer becomes the default answer.

Fournier's Distinction: Status vs. Coaching

Camille Fournier draws a sharp line in The Manager's Path between two things that look similar in a calendar but are functionally different: getting status and having a coaching conversation.

Status is information transfer. Project is at 73%. Design review is Thursday. Staging went down and came back up. Useful data. But it doesn't require a 1:1 — it requires a Slack message or a project management tool or a stand-up. When you use a 1:1 to collect status, you're wasting the meeting's highest-value property: the private, synchronous, relationship-rich channel it creates.

Coaching is something else. It's helping someone think through a problem they haven't fully solved. It's asking what they've tried, what they think the obstacle actually is, what a good outcome would look like, and — crucially — it's doing most of this through questions rather than solutions, so the person builds their own reasoning rather than borrowing yours.

Fournier is clear that managers often default to status collection because it's lower-stakes. You ask, they tell, you have information, the meeting ends, nobody had to get uncomfortable. Coaching requires you to sit with ambiguity. It requires you to not fix things. It requires the manager to develop what amounts to a tolerance for open loops — for leaving a conversation with "I'm not sure, let's both think about it and come back to it next week" rather than closing everything down.

The tell is usually the first question you ask. "Where does the project stand?" is a status question. "What's the hardest part of this project right now?" is a coaching question. Both take five seconds to say. One opens something, one closes it.

This doesn't mean you never collect status in a 1:1. Sometimes a quick project pulse-check is genuinely necessary context. But Fournier's suggestion — one worth taking seriously — is that you get into the habit of noticing which mode you're in, and choosing it deliberately. Most status can be captured async. Most

coaching cannot.

The Quiet Revolution (Ryan's Story)

When Ryan joined the team, the 1:1 calendar was already set up. Thirty minutes, every week, eight direct reports, all inherited from his predecessor. He assumed this was a good thing — one less logistical task in a first month that already had too many.

The first week, he ran them straight. Asked people where their projects stood. Took notes. Smiled. Said "great, thanks for the context." Was thanked in return.

The second week, he realized something was off. He couldn't actually tell you, from those first eight conversations, what anyone found genuinely hard about their work, what they wanted out of the next year, or whether anyone was unhappy. He had a spreadsheet's worth of project data and zero relationship data. He had heard eight versions of "things are fine" and had no way of knowing how many of them were true.

He went back and looked at his notes. Nearly every question he'd asked was a status question. Nearly every answer was a status answer. He'd run eight thirty-minute briefings and called them one-on-ones.

The third week, he tried something small. He stopped talking first. He sent each person the same message the day before: "I want to make our 1:1s more useful for you. Can you come in with one thing you want to talk through, one thing that's been bugging you, and one thing you're proud of this week?" Then he showed up, asked them to start, and kept his mouth shut for the first ten minutes.

The difference was immediate and, for two people in particular, jarring. One engineer — he'd barely exchanged a word with her outside of status updates — spent twelve minutes describing a technical disagreement she'd been having with a colleague that had been simmering for three months and was starting to affect her work. She'd never raised it because, she said, "I didn't think that was what 1:1s were for."

It is what 1:1s are for. Exactly.

A second person, a senior engineer with a quiet, self-contained manner, used his new opening to mention, almost offhand, that he'd been talking to a recruiter. Not as a threat. Just as information. He was interested, but he wanted to know if the things he wanted to grow into were available at this company. It was the first career conversation he'd had with anyone in his management chain in two years.

Within six weeks of that small change — stop talking first, send the prompt the day before, ask the coaching question before the status question — four conversations had meaningfully shifted. The team wasn't transformed. No dramatic turnaround. But Ryan had moved from managing projects to managing people, and the 1:1 was where the difference showed up first.

The Practical Mechanics (Which Still Matter)

None of the above works without some basic infrastructure. A few things that are worth getting right:

Frequency. Weekly is the default and usually correct for most direct reports. Bi-weekly works for very senior people who have good self-management and established trust with you. Monthly is almost never enough — by the time you meet, the issues have either resolved or calcified. If someone works remotely and your primary async channel is text, lean toward weekly as a counterbalance.

Never skip it. This is Grove's rule and it's unambiguous. When you cancel a 1:1 repeatedly — even for legitimately good reasons — the signal your direct report receives is that they are lower-priority than whatever bumped them. Once or twice for genuine crises is human. As a pattern, it damages trust faster than almost anything else a manager does.

A shared doc, not memory. Keep a running document for each 1:1 relationship. Both of you can add to it. Agenda items live there. Decisions and commitments from previous weeks live there. This does several things: it makes follow-through visible (people notice when you actually did the thing you said you'd do), it creates continuity across weeks so conversations can build on each other, and it means you're not starting from scratch every Tuesday.

Location flexibility. Walk-and-talk 1:1s work surprisingly well for certain kinds of conversations — they reduce the formality and the face-to-face pressure that makes some people clam up. If your office setup allows it and the person is in-person, suggest it occasionally.

Skip-level awareness. As your team grows and you start managing managers, your direct 1:1s are not your only window into the broader team. But they're still the most important one. Don't let the 1:1 with your managers become exclusively a meta-conversation about their reports — that's a status report with extra steps.

What to Do When It Isn't Working

Sometimes you do everything right and the 1:1 still feels hollow. The person gives short answers. The check-in question lands flat. You've asked the coaching question three weeks in a row and gotten variations of "no, I think I'm good."

A few things can cause this:

The relationship isn't there yet. Trust is built slowly, and some people don't open up in professional settings until they're convinced you're actually interested and not going to use what they tell you against them. Give it time. Keep showing up. Keep asking the real questions, even when the answers are surface-level. The consistency is itself a signal.

They've had bad managers before. This is more common than you'd expect. Someone who spent two years in 1:1s that functioned as performance check-ins — where candor occasionally produced consequences — has learned that the safe move is to keep it light. You can't fix this in a meeting. You fix it by being the kind of manager, over time, where the safe move is being honest.

The format needs adjusting. Some people are not well-served by open-ended questions in a synchronous setting. Try an async prompt the day before: "Before our 1:1, drop two or three things you want to talk about." The written preparation sometimes surfaces things that don't come up in the moment.

Something is actually wrong. If a person who used to be engaged in 1:1s has become distant and monosyllabic, that's a signal worth taking seriously rather than attributing to introversion or a bad week. Is there a project that's going badly? A relationship on the team that's broken? Something in their life outside work? The "what's been most draining lately" question, asked with genuine curiosity rather than managerial routine, sometimes opens this. Sometimes you have to just say: "I've noticed the last few weeks have felt quieter — is there something going on you want to talk about?"

The worst response to a 1:1 that isn't working is to let it drift back toward status updates because at least that's comfortable. That's how you end up, two years later, handing over a team to someone new — someone who will sit through eight thirty-minute briefings in their first week and have no idea what they've actually inherited.

The Meeting You Can't Get Back

Here's the frame that finally made 1:1s click for Ryan — and it might be the most useful thing in this chapter:

The 1:1 is the only regularly scheduled time you will have with each person on your team where the explicit purpose is them. Not the product, not the roadmap, not the incident postmortem. Them. Their work, their obstacles, their growth, their state of mind.

You have maybe forty-five working years. The people who report to you right now are spending a meaningful fraction of their lives doing work that you are, in some real sense, responsible for shaping. The 1:1 is where you either honor that or you don't.

That might sound heavier than a meeting deserves. But consider the alternative — the manager who treats it as a status ritual, who never finds out what anyone actually wants, who loses three good people in two years to managers who bothered to ask. The 1:1 doesn't require you to be a therapist or a career counselor or to have wise answers to hard questions. It requires you to show up, give the meeting to the other person, and ask something real.

Stop talking first. Send the prompt the day before. Ask the coaching question before the status question.

That's it. That's the whole revolution.

CHAPTER IV

Giving Feedback That Actually Lands

Here's something that will happen to you, probably within your first ninety days as a manager. Someone on your team will do something that needs to be addressed. Maybe they interrupted a colleague three times in a sprint review. Maybe they shipped code without running it by QA, again. Maybe they've been arriving forty minutes late to every stand-up and it's affecting morale in ways you can feel but haven't yet quantified.

You will sit on it for a week.

Then two weeks.

By week three you will have constructed, in your head, a very reasonable and detailed case for why now is not quite the right time to bring it up — the project is at a critical stage, they've had a rough month, you're still building trust, there's a lot going on. And then one Tuesday afternoon the issue will surface in a way you can no longer ignore, and you will call a meeting, and you will say things you haven't practiced, and it will go badly. Not catastrophically, usually. Just badly enough that you will lie awake afterward mentally transcribing the conversation and identifying seventeen things you should have said differently.

That's the feedback problem. Not what you say. When you say it, how you say it, and what kind of relationship was already in the room when you opened your mouth.

Why Most Feedback Fails Before You've Said a Word

Andy Grove had a blunt framing for the manager's job: your output is the output of the people you manage. Not your code, not your slide decks, not your clever thinking. Theirs. Which means that anything you do — or fail to do — that leaves a person on your team performing below their capability is a direct hit to your own performance. Feedback isn't a kindness you extend to your reports. It's your job.

The gap between knowing that and actually doing it well is where most new managers live for their first couple of years.

Here's what the research consistently shows: feedback fails most often not because the content is wrong but because it arrives too late, too vaguely, wrapped in the wrong emotional register, or inside a relationship that hasn't been built to hold it. A 2019 study by the NeuroLeadership Institute found that vague, general feedback triggered a threat response in recipients — elevated cortisol, reduced working memory, defensive cognition — nearly as reliably as no feedback at all. In other words, telling someone they "need to work on their communication" isn't just unhelpful. It's actively counterproductive. The brain hears "something is wrong with you" and goes to work defending itself rather than changing.

Julie Zhuo, writing about her early years as a manager at Facebook, is precise about this: "Feedback is only as useful as the action it can produce." If the person receiving feedback can't identify a specific behavior to change, the conversation produced feelings, not results. Usually defensive feelings.

So before we get to any model or framework, sit with this: the point of feedback is behavior change. Not the manager's catharsis. Not the completion of a responsibility. Behavior change. Everything else follows from that.

The Three Feedback Archetypes That Don't Work

There are roughly three ways first-time managers deliver feedback badly. You might recognize yourself in one of them. Possibly more than one, depending on the day.

The Avoider

The avoider knows exactly what needs to be said. They've known for two months. They've had three conversations with themselves about how to frame it. They've drafted a Slack message and then deleted it. They will bring up the issue, they tell themselves, after the current project wraps up.

The current project never quite wraps up.

What the avoider is doing, in the short term, reads as kindness or patience. In the medium term it's neither. Issues that aren't addressed tend to calcify. The behavior that needed a five-minute conversation in week one becomes a team-wide morale problem by month three. And by the time the avoider finally says something — usually under duress, usually when something has snapped — the feedback arrives freighted with all the accumulated frustration of weeks of watching and saying nothing. The person on the receiving end feels ambushed. They have no memory of establishing a pattern, because no one ever named the pattern.

Camille Fournier is clear on this in The Manager's Path: "Avoid creating a feedback debt." Every time you notice something and don't say it, you're borrowing against a conversation that will eventually be more expensive, not less. The interest rate compounds faster than you'd like.

The Lecturer

The lecturer has no shortage of things to say. Feedback, for the lecturer, is an opportunity to deploy analysis. They've thought about the situation carefully, they have a theory about what's wrong, and they're going to share all of it.

A feedback conversation with a lecturer runs about twenty-five minutes. The manager talks for twenty of them.

This fails for a reason that feels counterintuitive: you cannot talk someone into understanding themselves. Insight — the kind that produces real behavior change — tends to arrive when the person being given feedback reaches it, not when it's handed to them in a PowerPoint structure. A manager who delivers a lecture

may feel like they've been clear, thorough, and fair. The direct report, who has been trying to get a word in for fifteen minutes, mostly feels managed.

The other problem is that lecturing signals distrust. Implicitly, you're telling someone that they couldn't possibly have thought about this on their own. That they need you to explain their situation to them. Most people, especially senior contributors who were hired for their judgment, find this mildly insulting. They shut down. They stop engaging. They smile and nod and leave your office with exactly zero intention of changing anything.

The Sandwich Practitioner

You know this one. Positive, then corrective, then positive again. "You've been doing really great work on the API documentation — I did want to mention that the last three releases had some issues with the testing protocols, but overall your contributions this quarter have been really valuable and we're lucky to have you."

The theory behind the feedback sandwich is that you cushion the blow. The reality is that you dissolve the message. Researchers at Kansas State University found that recipients of sandwich feedback were significantly less likely to accurately identify the core corrective message than recipients of direct feedback — because the brain, which is alert to praise, gets distracted by the bread. The correction ends up as a mumbled middle clause in a sentence about how great everything is.

There's something else the sandwich does that's subtler and worse. It trains your team not to trust your positive feedback. Once people learn that "you've been great" is often followed by "but," they start bracing whenever you say something nice. The praise becomes a tripwire rather than genuine recognition. You've now made both your positive and your corrective feedback less effective at once.

The Model: Behavior, Impact, Ask

The structure that consistently outperforms the alternatives is not complicated. Three parts.

First: a specific behavior. Not a trait, not a pattern, not a vibe. A behavior. Something observable. "In the sprint review on Thursday, you interrupted Marcus twice while he was presenting the new onboarding flow" is a behavior. "You can come across as dismissive" is a trait. Traits are things people feel, argue about, and eventually dismiss as matters of opinion. Behaviors are things that happened. They're harder to argue with, and more importantly, they're actionable. A person can't stop "being dismissive" — it's too abstract to do anything with. They can stop interrupting people mid-sentence. That's a change they can make on Monday.

Second: the observed impact. What actually happened as a result of that behavior? Be specific here too — not "it affected team dynamics" but something like "Marcus lost his train of thought and didn't finish his point, and I noticed three other people on the call go quiet for the rest of the session." Impact grounds the behavior in something beyond your personal preference. It answers the implicit question every direct report has when receiving feedback: why does this matter? Without a clear impact, feedback feels like a manager managing for the sake of managing. With one, it connects the behavior to the outcomes you both care about.

Third: a forward-looking ask. Not a verdict. Not a punishment. A specific request for what's different going forward. "Next time you want to jump in during someone's presentation, can you wait until they've finished and use the Q&A?" Or sometimes just a question: "What would you need to be able to pause before responding in those moments?" The ask opens a conversation rather than closing one. It signals that you believe the person can change — which matters more than people think.

That's it. Three parts. What happened, what it cost, what you'd like to see differently. You can hold it in your head without note cards.

What makes this work isn't its elegance. It's the discipline it enforces. By requiring a specific behavior, you stop yourself from delivering feedback about someone's personality. By requiring a real impact, you stop yourself from complaining about things that don't actually matter. And by asking instead of instructing, you stay in a conversation rather than turning it into a deposition.

The Thing Under the Model

Here's what the model doesn't tell you, and what took me longer to understand: feedback is a bet you're placing on the quality of your relationship. The best-designed feedback in the world lands differently depending on what the other person believes about your intentions.

Julie Zhuo puts it plainly: "Before you can give feedback effectively, the person on the receiving end has to believe, genuinely believe, that you're in their corner." That's not a sentiment. It's a precondition. If your direct report thinks you're gunning for them, are threatened by them, or simply don't care about their success, your feedback — however well-structured — will be received as evidence of those beliefs. They're not hearing the content. They're reading the subtext.

This is why timing matters so much more than most managers are told. Feedback given immediately after an event is not the same as feedback given a week later in a scheduled meeting. The former is still connected to a shared context — you both remember what happened, why it happened, what the stakes were. The later conversation is abstract. It's a case file. And case files feel like prosecutions.

Andy Grove is direct about immediacy in High Output Management: feedback should happen as close to the triggering event as possible, before the context degrades and before the person has constructed a settled narrative about their own behavior. A quick "hey, can we talk about the meeting for five minutes?" the same afternoon is almost always better than waiting to "address it properly" in next week's 1:1.

When Ryan Blew It: The Marcus Story

The senior engineer's name was not Marcus. But the conversation was real, and it was the one that finally made me understand what I'm writing here.

He'd been on the team for three years before I became his manager. Smart, experienced, very aware of both. In the sprint reviews he tended to dominate — talking over junior engineers, finishing their sentences when

he thought they were taking too long, occasionally rolling his eyes at estimates he found optimistic. None of it was malicious. It was just his tempo, and nobody had told him it was a problem.

I let it go for six weeks because I thought I needed to "get to know him first." Which was true, but I also let it go because he was intimidating to give feedback to. He was older than me by eight years, had been at the company longer, and would almost certainly have opinions about being managed by someone who was, to his eye, still figuring things out.

When I finally brought it up, I did it in a 1:1, which was the right venue. I started with some genuine positive framing, which was the wrong move — he could feel the "but" coming and his posture changed. I then said something like "I've noticed that sometimes in sprint reviews the more junior engineers don't quite get to finish their thoughts, and I think that's something worth being mindful of." I used "sometimes." I used "might" twice. I said "worth being mindful of," which is the management equivalent of clearing your throat and leaving the room.

He said, "Okay, I hear you," and moved on to ask about a technical decision he wanted to discuss.

Nothing changed. Of course nothing changed. I hadn't told him anything specific. I hadn't described an impact. I hadn't asked him for anything. I'd performed feedback without delivering it, which is its own special failure mode.

The follow-up conversation was harder, because now I had to explain that the first conversation hadn't worked, which required me to admit that I'd delivered it badly, which required me to take some accountability in front of someone who I was worried already didn't quite respect my authority. That's an uncomfortable set of moves.

But that second conversation — the one where I came in with a specific example ("In last Tuesday's sprint review, you talked over Jordan twice while she was explaining the performance regression"), a real impact ("She stopped presenting and I didn't hear from her again the rest of the meeting, and afterward two other people mentioned it to me separately"), and an actual ask ("What I'd like is for you to let engineers finish their thought before you respond, even if it takes longer than feels comfortable") — that one landed. He was quiet for a moment. Then he said, "I didn't realize it was that visible." Then we talked for twenty minutes about where that habit came from and what he wanted to do about it.

The content of the second conversation wasn't that different from the first. The delivery was completely different. And the relationship had also shifted — because I'd come in having done the work, and he knew it.

Positive Feedback Isn't a Consolation Prize

There's a tendency among managers who've been burned by the sandwich to overcorrect — to become stingy with positive feedback in order to prove they're not softening a blow. This is its own mistake.

Positive feedback is not the same as praise for the sake of praise. Specific, timely positive feedback tells someone exactly what's working so they can do more of it. "The way you structured that client presentation

— leading with the risk analysis before the solution — that's exactly the right order for that audience and it showed" is not a sandwich opener. It's useful information about what good looks like, delivered close to the moment it occurred.

Camille Fournier makes the point that positive feedback is also how you calibrate your team's understanding of quality. If you only give corrective feedback, people learn what not to do. They don't learn the specific behaviors that constitute excellent performance on your team. That's a real gap — particularly for new employees who are still triangulating.

The discipline here is the same: specific behavior, observed impact, and some version of "keep doing this." It takes thirty seconds. It's worth thirty seconds.

The Feedback Conversation You Want to Avoid

One last thing. There is a version of this that managers sometimes drift into when they're frustrated, tired, or have waited too long: the accumulated-evidence conversation. This is when you come in with not one example but five, and you lay them out in sequence to prove a pattern.

It feels logical. You're being thorough. You're showing that this isn't a one-time observation. The problem is that the person on the receiving end experiences it as an ambush backed by a dossier. Instead of engaging with the feedback, they start disputing the evidence. Did that really happen the way you're describing it? Was that really what I said? Were those really three separate incidents or was there context you're missing?

You have now turned a feedback conversation into a fact-finding tribunal, and nothing useful will come of it.

One specific example, well-chosen, is almost always more effective than five. You don't need to prove a pattern with volume. If the pattern is real, the single example will prompt them to recognize it themselves. If they genuinely don't recognize it and seem blindsided, that's information too — maybe it's not as consistent as you thought, or maybe the behavior is less visible to them than it is to everyone else, which is itself a starting point.

One example. Specific. Impact. Ask. And then stop talking. Let them respond.

Before the Next Hard Conversation

Here's a practical step before you deliver the feedback you've been avoiding. Write it out.

Not to read from — just to find out if it's actually specific. If you write "the behavior" and you find yourself writing something like "tends to be dismissive" or "sometimes seems disengaged," you don't have feedback yet. You have a feeling. Keep working until you have a sentence about something that happened on a specific day. Once you have that, you're ready.

Then check the impact. Ask yourself: what actually happened as a result? Not what might happen if this continues, not what it suggests about their future trajectory, but what actually happened in the moment or in

the following days. If you can't name an impact, it's worth asking whether this needs to be a feedback conversation at all, or whether it's a preference of yours that you're treating as a performance issue.

Then decide what you're asking for. One thing. Specific and doable. Not "be more aware of how you show up in meetings" — that's a fortune cookie. Something they can try on Monday.

Write those three things down. Look at them. That's your conversation.

Grove's underlying argument in High Output Management — the one that runs under all the operational specifics — is that the manager's job is to train and motivate toward peak performance, and that feedback is the primary mechanism for doing that. Not performance reviews. Not promotion conversations. The ongoing, timely, specific feedback loop that happens between those formal milestones. The annual review doesn't change behavior. The Tuesday afternoon five-minute conversation does.

Most feedback conversations aren't comfortable. They're especially uncomfortable early in your management career, when you're still figuring out your authority, still wondering whether people will take you seriously, still calculating how much social capital you have to spend. That discomfort does not go away completely. But it does get smaller as you practice, and what you'll find — what I've found, what every manager I've talked to has found — is that the people on the receiving end of clear, specific, caring feedback respect it. Even when it's hard to hear. Especially when it's hard to hear.

The alternative — watching someone struggle, staying quiet, waiting for a moment that never arrives — isn't kindness. It's avoidance with better PR.

Say the thing. Say it clearly. Say it soon.

CHAPTER V

Receiving Feedback When You're the New Authority in the Room

There is a specific kind of silence that settles over a room when someone on your team is about to tell you something they're not sure you can handle.

You've probably already felt it. The slight hesitation before a direct report says "can I give you some feedback?" The way they frame criticism with so many qualifiers that the actual point gets buried under a foot of bubble wrap. Or — and this is the more common version — the way they don't say anything at all, because they've quietly decided that giving you honest feedback isn't safe.

That silence is a signal. And learning to hear it, and then do something useful with it, is one of the harder skills new managers have to build. Mostly because nobody mentions it during the promotion conversation.

Here's what typically happens instead. You get promoted. Your identity shifts. You are now, officially, The Manager — the person with the title, the performance review authority, the salary visibility, and the Confluence page where people log their concerns. Within about three weeks, you start to notice that certain conversations feel slightly different. People are more careful around you. More considered. Some of it feels like respect. Some of it is, in fact, people deciding what not to say.

The problem with authority — even modest, team-level authority — is that it reshapes the information that flows toward you. Your team has things they want to tell you. Useful things, true things, things that would make you better at your job. And some percentage of them are calculating whether telling you is worth the risk.

The Instinct You Need to Override

Julie Zhuo, in The Making of a Manager, names manager defensiveness as the single biggest killer of team psychological safety. Not lack of one-on-ones. Not poor feedback delivery. Not bad hiring. Defensiveness. The moment a manager hears something critical about themselves and reaches for justification instead of curiosity — that's the moment the team learns to stop being honest.

It's worth sitting with that for a second. Because defensiveness doesn't feel like a choice. It feels like self-preservation.

The first time someone told me something genuinely critical about my management — it was a senior engineer on my second team, about three months in, who told me during a 1:1 that I had a habit of making decisions in hallway conversations and then presenting them as settled — my brain immediately went to work on the rebuttal. But I was just thinking out loud. Those weren't real decisions. You're misreading the

situation. I didn't say any of that. What I said was "Thanks for bringing this up — can you give me an example?" And that pause, that tiny redirect, is about the only thing I did right in that conversation. The rest I spent quietly constructing an internal case for why he was wrong.

He wasn't wrong.

The instinct to defend yourself when someone criticizes how you're leading them is completely normal and almost always counterproductive. You need to understand why it's counterproductive, not just accept the directive to stop doing it, because understanding the mechanism is what makes it possible to interrupt.

When you defend yourself in response to feedback, you teach your team two things simultaneously. First: that feedback is a confrontation rather than information. Second: that you are not safe to be honest with. Those two lessons are deeply durable. People don't forget them after one iteration. They generalize from the single incident — that time I told Sarah she was doing something that wasn't working and she got defensive — to a policy. Don't tell Sarah things that aren't working.

Amy Edmondson's research at Harvard Business School, which has replicated across industries including a famous 1999 study on hospital teams, shows that psychological safety — the team's collective belief that it's safe to speak up, take risks, and be honest — is one of the strongest predictors of team performance. It's not the only predictor. But it's the one that's most easily destroyed by a single bad response to a single piece of feedback. You can spend months building an environment where people feel comfortable being candid. You can wreck most of that work in one defensive reaction.

So the first thing to internalize is this: your response to feedback is not just personal. It's policy. Your team is watching what happens when someone is honest with you, and they're updating their model of whether honesty is safe here.

Why Your Team Won't Just Tell You

Before we get to the mechanics of soliciting feedback, let's be honest about the structural problem you're up against.

Your direct reports have good reasons to be careful.

You write their performance reviews. You have input into their compensation. You decide, in a real and immediate way, whether their work gets recognized, whether they get the interesting projects, whether their promotion case gets made. Even if you are the warmest and most psychologically safe manager alive, the power differential is real. It shapes what people tell you and how they tell it. It can't be wished away.

Camille Fournier puts it plainly in The Manager's Path: the transition from peer to manager is almost never clean, and the people you manage are navigating a new relationship with someone who used to be on the same rung of the ladder they're still climbing. They don't fully know you yet — the manager version of you, anyway. And they're not going to volunteer uncomfortable truths until they have enough evidence that doing so won't backfire.

This means the burden of creating conditions for honest feedback falls almost entirely on you. Waiting for your team to spontaneously offer critical upward feedback is a losing strategy. The feedback you want most — the feedback about your blind spots, your communication patterns, the way you run meetings, the way you deliver bad news — is exactly the feedback people are least likely to offer without an explicit invitation and a demonstrated track record of safe receipt.

You have to go get it. Actively, and more than once.

How to Actually Solicit It

There are three useful formats, and you'll probably end up using all three at different times.

The direct ask in 1:1s. The easiest place to start is the one-on-one you're already running. Toward the end of the meeting, add a question. Not "do you have any feedback for me?" — that question is almost a trap, because the only socially comfortable answer is "no, you're great." The better questions are specific and behavioral. "Is there anything about how I run our team meetings that makes them harder to get something out of?" or "When I give you feedback on your work, does it feel actionable, or does it land differently than that?" or the blunter version Zhuo recommends: "What's one thing I could do differently that would make your job easier?"

Those questions put a concrete frame around the ask. They signal that you have specific behaviors in mind that might be adjustable, not just a vague interest in being praised. They make it easier to answer honestly because the question itself normalizes the possibility that something needs to change.

When you get an answer, shut up. Don't fill the silence. Don't rush to context or explanation. Write it down in front of them so they can see you're taking it seriously.

The anonymous survey. For teams where trust isn't fully established yet — or where you're dealing with a particularly sensitive topic — a short anonymous survey is a legitimate tool. Three to five questions, kept narrow, and actually reviewed in detail. "What's one thing I do well as a manager?" and "What's one thing I should do differently?" as a paired set is almost always more generative than a longer form. Tools like Google Forms or Lattice work fine. What matters is that you actually look at the responses and then do something visible with them.

The common mistake with anonymous surveys is running them and then going quiet. If you ask your team anonymously what you're doing wrong and then don't mention it again, you've made it worse. They know you read it. The silence is its own message.

The 360 or skip-level. At some organizations, formal 360-degree feedback is built into the performance cycle. If yours isn't, you can approximate it informally by asking your own manager for observations, and occasionally asking a peer manager you trust whether they've heard anything from your team indirectly. This isn't gossip-mining — it's a legitimate way to access the feedback that travels sideways in organizations rather than vertically. Your team might not tell you directly that your sprint planning meetings run long and

people are frustrated. They might tell each other. They might tell the project manager who attends those meetings. You want access to that information.

Andy Grove, in High Output Management, frames information-gathering as a core part of the management job — not a passive receipt of formal reports but an active, ongoing effort to know what's actually happening at the working level. That's as true for information about your own performance as for anything else you're managing.

What to Do When It Stings

Here's the scenario you need to be ready for. You ask. Someone tells you something real. It's uncomfortable. Maybe it's that you play favorites in who gets the high-visibility projects. Maybe it's that you're inconsistent — you say one thing in team meetings and another in hallway conversations. Maybe it's that people feel like you're always rushing through 1:1s. Maybe it's something even harder, something that implicates not just a behavior but something closer to your character.

It stings. Your brain starts generating the rebuttal.

The next thirty seconds matter more than almost anything else.

What you do not do: argue. Even if you think they're wrong. Even if the feedback is based on a misunderstanding that you could clear up in two sentences. The moment you explain why their experience of your behavior isn't accurate, you've confirmed the thing they were worried about when they decided whether to tell you. You've made it a debate instead of information.

What you do: say thank you and mean it. Ask one clarifying question if you need more specificity to act on the feedback — not a clarifying question that is actually a rebuttal in disguise, but a genuine one. "Can you give me a specific example?" is usually the right move. It helps you understand the feedback more precisely, and it signals that you're trying to understand rather than dismiss.

Then you stop. You sit with it. You give yourself the time — a day, sometimes a week — to separate the sting from the substance.

Because here's what's almost always true: there's something real in it, even when it's imprecisely delivered. Even when the person giving the feedback has their own motivations or blind spots. The feedback that hurts the most is usually the feedback that's pointing at something you already knew, at some level, but had been managing not to look at directly. Defensiveness is often just the friction of a belief being updated.

After the 1:1 where my senior engineer told me I was pre-making decisions in hallway conversations, I spent about four days being privately annoyed at him. I ran through a mental catalog of examples that contradicted his point. I told myself the story of how I was actually a collaborative decision-maker and he just hadn't been paying attention to the right meetings. Then I went back and looked at my Slack history for the previous three weeks.

He was right. I had been doing the thing. Not always and not maliciously, but with enough consistency that someone paying attention would see a pattern.

What I had to do next was harder than just accepting the feedback intellectually. I had to change the behavior. And I had to do it visibly enough that he — and the rest of the team — could see that the feedback had actually moved something.

Closing the Loop in Public

This is the part most managers skip, and it's the part that actually builds trust over time.

When your team gives you feedback and you do something with it, they need to know that what they said is what changed the behavior. If you quietly adjust without saying anything, the connection between their honesty and your response stays invisible. The next time someone is deciding whether to give you feedback, they don't have evidence that doing so works. You have to make the loop explicit.

There are two ways to do this, and you'll probably use both.

The first is direct acknowledgment in the next 1:1 or team forum. "Last week, [name] told me that I tend to signal decisions in hallway conversations before the team has had a chance to weigh in. They were right, and I'm going to try doing X differently going forward. If you notice me slipping back, I want to hear about it."

That's it. That's the whole move. It's uncomfortable to say out loud, especially the first few times. It also demonstrates, unmistakably, that giving you feedback is not a ceremonial act — it's a causal one that actually shifts how you operate.

The second is what Zhuo calls the manager's public accountability — making your response to feedback part of how you talk about your own development as a manager. Not in a performative way, not as a recurring exercise in self-flagellation, but as a genuine model of what it looks like to be someone who takes information in and adjusts. When your team sees you doing that, they start doing it too. Psychological safety, it turns out, is contagious in both directions. If the person with the most authority in the room can receive feedback without getting defensive, it becomes a lot easier to imagine that everyone else can too.

The Specifics of Harder Feedback

Not all feedback is behavioral and adjustable. Some of it is structural. Some of it is about the company or the organization, not you. And some of it — the kind you most need to be careful with — is about your judgment, your decisions, or your fundamental approach to the job.

When someone tells you that they think you made the wrong call on a product decision, or that they don't understand your strategy, or that they feel like their work doesn't matter — that's not the same as being told you run long meetings. It requires a different kind of response.

Start by separating the feedback from the implicit question underneath it. "I don't understand the strategy" often means "I don't feel included in the thinking" or "I don't see how my work connects to what matters." That's addressable, and addressing it isn't about defending your strategic judgment, it's about better communication. "My work doesn't matter" often means "Nobody tells me whether I'm doing well" — which is a failure of recognition and feedback flow that you can fix.

This is where Fournier's framing in The Manager's Path is useful. She distinguishes between the manager's job and the individual contributor's job in terms of what each person is accountable for. The manager is accountable for creating conditions in which the work can get done well. When someone on your team is suffering from unclear priorities, opaque decision-making, or poor information flow — those are conditions. They're your conditions to improve. Receiving that feedback as an attack on your character is missing the point. Receiving it as information about the environment you're creating is the right frame.

The Feedback You Won't Hear

Here's the honest part.

Even if you do all of this well — even if you ask good questions, receive feedback without flinching, close the loop publicly, and build a team culture where honesty is genuinely safe — there will still be feedback you never hear. Some of it will leave with the people who go work somewhere else. Some of it will stay in private messages between your team members. Some of it will be the kind of thing that's just too uncomfortable to say to the person's face, even in the best conditions.

You can't solve for all of it. What you can do is make the gap as small as possible.

The skip-level conversation — a meeting your manager has with your direct reports without you present — is one of the more useful mechanisms for catching what doesn't make it to you directly. Good organizations build skip-levels into the cadence for exactly this reason. If yours doesn't, it's worth asking your manager to do them anyway. Yes, it's slightly uncomfortable to invite that. It's less uncomfortable than staying blind.

Exit interviews are another. Most exit interviews are conducted by HR and produce data that sits in a spreadsheet somewhere. If someone on your team is leaving, ask for fifteen minutes with them yourself — not to litigate their decision, but to genuinely ask what they wish had been different. People who are already leaving have the least to lose from honesty. The feedback you get in those conversations is often the most unfiltered version of what others on your team have been thinking but haven't said.

The Longer Game

Being good at receiving feedback doesn't mean being a pushover. It doesn't mean accepting every piece of feedback as equally valid, or agreeing with every criticism someone raises. It means something more specific: it means you can hear hard things without shutting the channel down.

Some feedback will be wrong. Some will be based on incomplete information. Some will be a reflection of someone's frustration with a situation rather than an accurate read of your behavior. You get to disagree with feedback — but you do that privately, over time, not in the moment when your first instinct is to defend.

The practical test I've found useful: after I've had twenty-four to forty-eight hours to sit with a piece of feedback that my gut wants to reject, is there anything in it that's accurate? Not entirely accurate. Not perfectly framed. Just partially, uncomfortably true. In my experience, the answer is almost always yes. And "almost always yes" is enough to take the posture of curiosity seriously, even when the feedback is imperfect.

Grove's framing of the manager's role as someone who must maintain high information quality applies here. The information channel that tells you how you're doing as a manager is extremely easy to corrupt. Power corrupts it. Busyness corrupts it. The perfectly natural human desire to believe you're doing a good job corrupts it. You have to actively maintain that channel — through the questions you ask, the responses you give, and the way you're seen to handle the answers when they're hard.

What your team remembers, more than the feedback you gave them or the projects you ran or the processes you built, is how you handled being told something you didn't want to hear. Whether you showed them that honesty was safe. Whether you were the kind of manager who made the information flow better just by being in the room.

That's a reputational thing that compounds over years. But it starts in the first three months, in the thirty seconds after someone says something critical, when your brain is running the rebuttal and you have to decide what to do with it.

Say thank you.

Write it down.

Go figure out if they're right.

They probably are.

CHAPTER VI

Hiring: The Part Most Managers Fumble

The first time I had to fill a role, I did what most new managers do. I read the old job description — the one written when the position was created three years earlier, for a different team solving a different problem — added a handful of bullet points about being a "self-starter" who "thrives in a fast-paced environment," and posted it. Then I scheduled interviews with whoever looked interesting to me, which really meant whoever reminded me of myself at twenty-eight. I made the hire based on a gut feeling I dressed up with phrases like "strong culture fit" and "she just gets it."

The person I hired lasted eleven months.

She wasn't a bad hire exactly. She was a wrong hire — wrong role, wrong moment, wrong expectations on both sides. I had set her up to struggle and then spent nearly a year managing that struggle. The cost wasn't just hers.

Hiring is the highest-leverage thing you do as a manager. Andy Grove was not subtle about this. In High Output Management, he frames a manager's output as the output of the organization they lead — the people on your team are the organization. Every other tool in this book, the 1:1 framework, the feedback model, the performance review template, is downstream of who you hire. Get the hiring right and the rest of the problems are manageable. Get it wrong and you're spending the next eighteen months either managing out a bad fit or quietly absorbing the gap they left when they burned out and quit. Estimates of the fully loaded cost of a bad hire — recruitment, onboarding, lost productivity of adjacent team members, the slow drag on morale — consistently land between one and two times the person's annual salary. And that's before you count the opportunity cost of the better candidate you didn't hire.

Most first-time managers treat hiring the way they treat buying a used car. They have a rough idea of what they want, they rely on gut instinct when they sit across from someone, and they rationalize afterward. This chapter is about replacing that instinct-dressed-as-process with something that actually works: writing a role definition that filters, building an interview loop that reduces the outsized role of gut feeling, and learning to sell your team honestly to candidates who have other options on the table.

Write a Role Definition, Not a Job Description

There is a difference between a job description and a role definition. Almost every company conflates them.

A job description is a wishlist. Twelve bullet points of required skills, half of which are aspirational, some of which contradict each other, most of which were copied verbatim from a similar posting somewhere else. You've read hundreds of them. They all sound the same because they're all descended from the same organizational bad habit.

A role definition asks three harder questions: What does success look like in ninety days? What are the two or three problems this person will genuinely own — not participate in, not support, but own? And what are the real constraints of the role, the things that would make someone wrong for it even if their resume looked perfect?

Start with the success criteria. Not "contributes to the team" or "delivers high-quality work" — those are circular. Write it like you'd write an acceptance test. In ninety days, this person has shipped the authentication refactor and documented the edge cases. In six months, they've established a working relationship with the design team that currently has no process. In a year, they're the person I'd trust to represent this team in a cross-functional planning meeting without preparation. That specificity is uncomfortable to write because it requires you to have thought hard about what the role is actually for. Most managers skip this step because it's hard. Do not skip this step.

Then write the actual constraints. Camille Fournier, in The Manager's Path, makes this point directly: the role you're hiring for is not a general concept, it's a specific person solving specific problems inside a specific team dynamic. If your team is four months from a major migration and will be in crunch mode, that's a constraint. If your most senior engineer leaves at the end of Q2 and this hire will need to absorb institutional knowledge fast, that's a constraint. If the role involves constant stakeholder negotiation with a difficult product partner, that's a constraint. Put it in the definition. It will scare off some candidates. Good. Those candidates would have struggled.

The job posting you put publicly can be a cleaned-up version of this — you don't owe the internet your team's internal dynamics. But the role definition lives inside your head and your hiring team's heads, and it governs every decision downstream.

One discipline worth forcing on yourself: separate the required skills from the nice-to-haves, and then cut the required list in half. Most requirements lists are badly inflated. The skills you genuinely cannot train on the job belong on the required list. The rest belong nowhere, or in a section labeled "you'll grow into this." Julie Zhuo makes the point in The Making of a Manager that managers consistently overspecify requirements, which has a direct effect on candidate pool diversity — particularly when qualifications like "ten-plus years of experience" get attached to roles where seven years of focused, relevant work would be more than enough.

Build a Structured Interview Loop

Here's the uncomfortable truth about unstructured interviews: they feel reliable and aren't.

Decades of personnel research have converged on this finding. In a landmark 1998 meta-analysis, Frank Schmidt and John Hunter reviewed eighty-five years of selection research and found that unstructured interviews — the kind where you ask whatever comes to mind and form an impression — predict job performance with a validity coefficient of about 0.38. Structured interviews, where every candidate is asked the same predetermined questions and rated against a defined rubric, come in around 0.51. That gap might sound modest until you multiply it across every hire you make in the next decade.

The reason structured interviews outperform gut-feel interviews is also the reason most managers resist them: they constrain you. You can't follow the interesting tangent. You can't spend twenty minutes talking about distributed systems architecture and then decide the candidate "gets it." You have to ask the same questions in roughly the same way and rate the answers the same way.

This is a feature. Not a bug.

Design your interview loop around the actual demands of the role. If the role requires influencing without authority — convincing other teams to prioritize your requests — design a question that surfaces that specifically. "Tell me about a time you needed something significant from a team that didn't report to you. Walk me through what happened." That's a behavioral question tied to a real requirement. "What's your greatest weakness?" is an invitation to a prepared monologue that tells you almost nothing.

Each interviewer in the loop should own a distinct signal. One person evaluates technical depth. One evaluates communication and collaboration. One evaluates the candidate's relationship to ambiguity — can they make good decisions without complete information? One evaluates their approach to failure and learning. The signals shouldn't overlap, because overlapping signals produce redundant data and wasted time. After the interviews, each person writes their assessment independently before the group debrief. This matters more than it sounds. Daniel Kahneman, Olivier Sibony, and Cass Sunstein document in Noise how powerfully the first opinion expressed in a group debrief anchors everyone else's judgment. Write it down first. Discuss after.

Your rubric doesn't need to be elaborate. A four-point scale — doesn't meet the bar, meets the bar, exceeds the bar, exceptional — applied consistently creates enough comparative data to make better decisions. The rubric should include two or three specific behavioral markers for each level. "Exceeds the bar on cross-functional influence" means: gave a concrete example with a real outcome, demonstrated understanding of the other team's incentives, showed initiative in the relationship rather than waiting to be engaged. Write those markers before the first candidate comes in.

Structure also forces you to confront what you're actually evaluating. Bias doesn't disappear with structure — Mahzarin Banaji and Anthony Greenwald's research on implicit association makes clear that we carry associations we can't fully access consciously — but structure makes it harder to act on bias invisibly. Marianne Bertrand and Sendhil Mullainathan's 2004 study sent identical resumes with different names to employers and found that resumes with white-sounding names received 50 percent more callbacks than identical resumes with Black-sounding names. The interviews didn't even happen; the bias operated at the screen. When your criteria are explicit — when you have written, agreed-upon definitions of what "meets the bar" means — it's at least harder for a candidate to fail for reasons that have nothing to do with the role.

Fournier is direct about the cultural dimension: "culture fit" is a phrase that should make you suspicious every time you hear it, including when you hear it from your own mouth. It usually means "feels like someone I'd be comfortable with," which often means "reminds me of the people already here," which is precisely how homogeneous teams stay homogeneous. Ask instead what specific values and working styles the role

requires. Evaluate those. Culture isn't a feeling. It's behavior.

The Reference Check You're Not Taking Seriously

Most hiring processes treat the reference check as a formality. Three calls, fifteen minutes each, everyone says the candidate was wonderful, offer extended. This is nearly useless.

The reference check becomes useful when you treat it as a structured interview. Call references the candidate didn't provide if you can — former managers from earlier in their career, skip-level colleagues, anyone who saw their work from a different angle. Ask specific behavioral questions rather than "what were their strengths?" Ask: "Tell me about a time they pushed back on a direction you gave them. What happened?" Ask: "If you were building a new team from scratch tomorrow, would you hire them — and for what kind of role specifically?" The specificity of the answer tells you more than the sentiment. Hedged praise and long pauses are data.

Sell Honestly, or You're Just Deferring a Problem

This is where I went badly wrong. I want to be specific.

A few years into managing, I was trying to hire a senior engineer — call her Priya — for a role I was genuinely excited about. The team was doing interesting work. The technical problems were real. I believed in the direction. When I talked to her about the role during our conversations, I talked about the interesting parts. The autonomy. The scale. The fact that she'd have real ownership of an entire system.

I didn't lie. But I didn't tell her that the codebase she'd be inheriting was carrying fourteen years of accumulated technical debt. I didn't mention that the product partner she'd work closest with had a habit of changing requirements at the end of a sprint. I didn't say that the "ownership" I was describing would take about eight months to fully materialize because an existing contractor was finishing a major component first.

She was exceptional in the interviews. We made the offer. She took a few days and then called me.

She said, gently but without ambiguity, that she'd spoken to a former colleague who'd worked at the company. The picture she had of the role was meaningfully different from the picture that conversation had painted. She turned down the offer. She was polite about it. She was right.

I spent the following month rebuilding the process. The most important change I made was this: I started telling candidates the actual truth about the role, including the parts that reflected poorly on where we were.

Here's why this works. Candidates who have real options — the ones you most want to hire — are evaluating you as much as you're evaluating them. They have been burned by oversold roles. They've taken jobs where the "exciting early-stage opportunity" turned out to mean no processes, no support, and a product direction that changed every six weeks. They are primed to detect gaps between the pitch and the reality, and when they detect one, they either decline the offer or they take it and leave at month eight. Neither outcome serves you.

The honest pitch sounds something like this: Here's what's genuinely compelling about this role. And here's what's genuinely hard. The codebase has debt. The product relationship has friction. The ownership is real, but it'll take time to earn in full. If those sound like challenges you'd want to take on, I want to make the case for why this team and this moment are worth it. If they don't, I'd rather know now.

Zhuo describes this as selling the real company rather than the imaginary one — the idea that the best hires happen when both parties have accurate information about what they're signing up for. The people who stay through a difficult quarter are almost always the ones who knew it was coming. The ones who leave at month seven are almost always the ones who were surprised.

Be specific about what "selling honestly" means in practice. In the pitch conversation — which should be separate from the evaluation interviews, ideally — cover three things: what's compelling about the team and the work, what's genuinely difficult about the current moment, and why you think this particular candidate is well-suited to the difficult parts. That last piece matters. It's not just "here are the challenges." It's "here are the challenges, and here's why I think the way you approached the last hard problem suggests you'd find these interesting rather than defeating."

There's also a practical dimension: your current team is watching how you hire. They're in the interview loop. They're having lunch with candidates. They hear how you talk about the role. If they see you overselling, they know you're doing it, and they're filing that information away about your credibility. Honest hiring is also team culture.

The Offer Isn't the End of the Decision

Most managers treat the signed offer letter as the conclusion. It isn't. The period between offer and start date is when candidates second-guess themselves, take other calls, and occasionally reverse course. Stay in contact. Not intrusively — nobody wants a daily check-in from their future boss — but enough that the offer doesn't feel like it came from a company that went silent the moment it had their signature.

And think concretely about the first two weeks. Does this person have a clear picture of what their first week looks like — who they'll meet, what they'll be working on, what "good" looks like in thirty days? The onboarding plan you write is part of hiring. Candidates who feel genuinely set up to succeed in their first month are less likely to be the ones you're managing out in month eleven.

The Math on Leverage

Grove's framing is worth sitting with. If your output as a manager is the output of your team, and the quality of your team is substantially determined by the quality of your hiring, then a mediocre hire has a compounding effect. Not just in the direct cost of their underperformance, but in the opportunity cost of the stronger candidate you didn't hire, the time your team spends working around someone who isn't pulling weight, and the slow signal it sends about your standards.

A hire that doesn't work is not a neutral outcome. It's a negative one that compounds.

The inverse is also true. The engineer who joins and immediately raises the quality bar — who writes documentation that makes the next three new hires more effective, who asks the questions in design reviews that catch problems early, who becomes the de facto technical conscience of the room — that person's impact is worth several times their salary on paper. The investment of structured time in hiring them properly pays back for years.

Before you write a word of the job description, answer these questions in writing:

What does success in this role look like at ninety days, six months, and one year? Write specific outcomes, not adjectives.

What are the two or three hardest problems this person will face? Are you prepared to tell candidates about them honestly?

What skills are genuinely required — meaning you'd reject a candidate who lacked them — versus simply desired? Cut the required list to what you actually can't train.

Who is in the interview loop, and what distinct signal does each person own?

Are you collecting individual assessments before the group debrief?

Have you prepared the honest pitch — the compelling parts and the hard parts?

These questions take maybe two hours to answer well. The hiring process they produce runs for weeks. The person you hire will work with you for years. The two hours are the cheapest investment in the entire sequence.

The gut feeling doesn't vanish. You'll meet candidates and feel something — a pull, or a hesitation you can't immediately articulate — and that instinct isn't worthless. It's one input among several. When it conflicts with your structured data, the data should usually win. When it aligns with the data, you have full permission to feel good about the decision.

But you don't get to skip the data. You're a manager now. This is the actual work.

CHAPTER VII

Firing: When, How, and the Dignity Question

Nobody tells you that firing someone will be the hardest thing you do as a manager. Not the most strategically complex, not the most technically demanding. The hardest. The kind that sits in your chest for days beforehand and doesn't leave your stomach for days after.

And because it's hard, most managers wait. Then wait longer. Then find a reason to wait again.

That waiting is where the real damage happens.

The Cost of Not Deciding

Here's what accumulates while you're building courage: your best performers watch. They always do. They see you tolerating the person who misses deadlines, deflects ownership, or poisons team meetings with a particular brand of low-grade negativity, and they draw conclusions. Those conclusions are rarely charitable. In the 2017 Corporate Executive Board study on team performance — one of the most cited in organizational behavior — the single biggest driver of high performer attrition was the perception that management let underperformance slide. Not compensation. Not growth opportunities. Watching someone get away with it.

Andy Grove put a sharper point on it in High Output Management: "When a person is not doing his job, there can be only two reasons for it. The person either can't do it or won't do it." That binary is bracing, but Grove's point wasn't cruelty — it was clarity. Every week you spend not naming which category you're in is a week you spend treating the wrong problem. And a week the rest of your team spends noticing.

The person being managed poorly loses too. This is the part that makes avoidance feel compassionate but isn't. Someone who's been underperforming for six months — who has received vague feedback, half-hearted coaching conversations, and no clear signal that their job is at risk — has been denied the information they needed to make adult decisions about their life. They've been set up for a shock that didn't have to be a shock. That's not kindness. It's a manager protecting their own discomfort at someone else's expense.

None of this means you should fire quickly, cavalierly, or without doing the work first. It means the decision deserves a real decision, not a slow fade.

The Decision Framework: Are You Actually There?

The question that traps most first-time managers isn't how to fire someone. It's whether they've honestly answered should they. The two failure modes are opposite: acting too fast, before genuine support has been tried, and acting too slowly, after the support has run its course and you're just going through motions you don't believe in.

Camille Fournier, in The Manager's Path, offers a test that I've found more useful than most: ask yourself whether you've told this person clearly and directly that their job is at risk. Not that you're "concerned about some patterns," not that you "wanted to flag some feedback." Whether they know — in plain language — that continuing on this trajectory ends their employment. If the answer is no, you're not ready to fire. You're ready to have the conversation that makes firing possible, if nothing changes.

That conversation is harder than the termination itself. Surprisingly. Because in the termination meeting you've already decided. In the "your job is at risk" conversation, you're still hoping.

The four questions I'd put to any manager who thinks they're at the threshold:

One. Have you told them specifically what needs to change and by when? Not impressionistically — specifically. "Your code reviews need to be done within 48 hours of the request, not five days, and the comments need to be substantive, not approvals with no feedback" is specific. "I need you to be more responsive and engaged" is not.

Two. Have you given them the resources to actually change? Training, mentorship, reduced scope, paired work — whatever was relevant. Firing someone for a skill gap you never tried to close is a management failure, not a performance failure.

Three. Have you documented this honestly, in writing, in real time? Not retroactively assembled, not in private notes you'd be embarrassed to show HR. In writing, shared with them.

Four. Has anything changed? If you've done one through three and the answer to four is no, you're there.

If you can't honestly answer yes to the first three, you're not at termination. You might be at a performance improvement plan. Those are different things.

The PIP: Genuine Tool or Elaborate Formality

The performance improvement plan has a reputation problem. Most people — employees and managers alike — treat a PIP as the polite precursor to a firing, a piece of documentation theater that protects the company legally while the exit paperwork gets drafted in parallel. Sometimes that's exactly what it is. When it is, everyone knows it, and the whole exercise corrodes trust in ways that outlast the individual's departure.

But a PIP doesn't have to be theater. Done honestly, it's a structured contract: here's what's not working, here's what success looks like, here's the support we're providing, here's the timeline, here's what happens at each checkpoint. The key word is honestly. If you're writing a PIP that you intend to actually follow through on — where the person could genuinely succeed and keep their job — the document looks different than one written to create a paper trail. The honest version has achievable targets. It has support mechanisms. It has interim check-ins that are real coaching conversations, not just documentation sessions.

The tell is this: if someone asked you, "If they hit every goal in this PIP, do they keep their job?" and the honest answer is "probably not" — you're not running a performance improvement plan. You're running a severance delay. That's a thing managers do. It's understandable. But call it what it is, at least internally, and

don't let yourself believe it's the same as genuine investment.

Julie Zhuo writes in The Making of a Manager that the question to ask yourself is whether you're rooting for the person to succeed. That's the diagnostic. If you're mentally calculating their last day while writing the PIP's success criteria, the compassionate move is to move faster and more honestly, not to extend the process.

When to use a genuine PIP: early-stage performance issues where the person has been unclear on expectations, new-to-role struggles that might be fixable with support, situations where you suspect the problem is circumstantial rather than dispositional. When not to use a PIP: after a serious integrity or conduct violation, when the role has fundamentally changed beyond the person's capacity, when the relationship has broken down to a degree that the work environment is genuinely harmful to the team.

The Legal Basics (Because You Can't Not Know This)

This chapter is not a substitute for talking to HR and, if necessary, your company's employment counsel. Every jurisdiction is different. Every employment contract is different. I'm going to give you the floor, not the ceiling.

In most U.S. at-will employment contexts, you can terminate employment without cause as long as the reason isn't discriminatory or retaliatory. That second clause matters. "Without cause" does not mean "for any reason whatsoever." Firing someone immediately after they filed an HR complaint, returned from FMLA leave, or disclosed a disability is legally dangerous regardless of whether there was also a legitimate performance rationale. Timing creates inference. Be aware of it.

Documentation matters for the same reason. Not because you should be building a legal case before every hard conversation — that's exactly the kind of adversarial posture that makes terminations awful — but because honest, contemporaneous records protect both parties. They protect the employee from a sudden unexplained firing. They protect you and the company from revisionism. Write things down close to when they happen.

Severance is usually not legally required in the U.S. for at-will employees, but whether your company offers it and what the terms are is something you need to know before you walk into the room. The employee will ask. You should have an answer.

One more thing: know before the meeting what happens to their access. Email, systems, Slack, physical building access. In many companies, IT disables accounts at the moment of termination. In some roles — especially roles involving sensitive data or customer relationships — that's the right call. In others, a short transition window is appropriate and more humane. Talk to your manager and HR about what the protocol is and why, so you're not improvising in the room.

The Conversation Itself

The termination meeting should be short. Fifteen to twenty minutes. This runs counter to the instinct of the conflict-averse manager, who wants more time to explain, soften, revisit, and give the person a chance to respond in a way that changes the outcome. The outcome doesn't change in this meeting. If it could, you're not ready to have the meeting.

A few practical absolutes:

Don't do it on a Friday. The person will spend the weekend unable to reach HR, their manager, or anyone who can answer questions, alone with their anger and anxiety. Monday or Tuesday. If they need to clear their desk and leave, better they have accessible support around.

Have HR in the room. Always. This is standard practice and exists for everyone's protection.

Have a script, at least for the first two minutes. You don't have to read from it. But know your opening sentence. Something like: "I have some difficult news. We're ending your employment, effective today." Not "We've decided to make a change," not "This has been a really hard decision," not a preamble of any kind before that sentence. The kindest thing you can do is say it first.

Then: what happens next. Severance, if any. COBRA continuation information. The timeline for collecting personal belongings. Their final paycheck. References — what you can and can't say.

What you don't do: relitigate the performance history. Justify the decision at length. Apologize for the decision while also defending it (this is a very common pattern and it's confusing and hurtful). Ask them how they're doing. Make promises you can't keep about references.

What you do do: let them feel what they feel. If they're angry, that's legitimate. You don't have to absorb it without limit, but you don't have to deflect it either. If they're silent, let the silence sit. If they cry, have tissues. This is a human moment. It doesn't require management. It requires presence.

Ryan's Two Stories

I got one right and one badly, instructively wrong.

The one I got wrong came first. About fourteen months into my first management role, I had a product manager on my team — I'll call her Dana — whose work had been sliding for two quarters. Missed deadlines. Specs that required too many revision cycles. A communication style that had started generating friction with the engineering team in ways I was quietly smoothing over instead of naming. I had conversations with her — real ones, I thought — about the patterns I was seeing. She acknowledged them, committed to changes, and the changes lasted about three weeks before things drifted back.

What I didn't do: tell her clearly that her job was at risk. I was using every softened formulation available. "I'm concerned about the trajectory." "I need to see some sustained change here." I told myself these were clear signals. They weren't. Not for someone who hadn't received explicit warnings before.

When I finally escalated to my manager — who then made the call to move toward termination — Dana was blindsided. She said, in the exit meeting, that she'd had no idea it was this serious. And the worst part was that she was right. I'd failed her in the feedback phase. The termination was almost a secondary injury. When she said she hadn't known, I didn't have a clean defense, because the defense would have required me to believe that my gentle formulations were as clear as I'd told myself they were. They weren't.

The one I got right — or at least got less wrong — was two years later. A senior engineer, call him Marcus, who had been struggling since a reorganization that put him in a cross-functional role that didn't play to his strengths. Six months of genuine work: repositioning the role, pairing him with a more senior architect, having the hard conversation explicitly and early — including the words "I need you to understand that if this doesn't improve in the next sixty days, we'll need to talk about next steps, and those next steps include the possibility that this role isn't the right fit." That sentence was hard to say. It's the kind of sentence that makes you feel like a person delivering a threat. But it's information. It's the information he needed to take the situation seriously.

He didn't turn it around. The PIP was genuine, the goals were achievable, the support was real. At the end of the sixty days we were in roughly the same place.

In the termination meeting, he said something that has stayed with me: "I knew this was coming. I appreciate that you were straight with me." It was not a warm conversation. He was disappointed and frustrated. But there was something clarifying about the fact that the ending matched the warning. That it wasn't a surprise. He left that company, found a role as an individual contributor at a smaller company that was a much better match for how he worked, and I ran into him two years later at a conference. We had a normal conversation.

The difference between Dana's story and Marcus's isn't that one was more fair or more legally sound. It's that in Marcus's case I did the work before the meeting that made the meeting honest. The termination was the last step in a transparent process. In Dana's, the termination was the first real signal.

After the Meeting: The Team

You will spend thirty minutes thinking about the termination meeting and three months managing what comes after. How you handle the team announcement is, in some ways, as important as how you handle the conversation itself.

Say something. The same day. Don't let it become a rumor that spreads unevenly through Slack. A short, factual message — "I want to let you all know that [Name] is no longer with the team, effective today. I can't share details, but I'm available to talk if you have questions" — is enough. It doesn't have to explain. It just has to confirm.

Expect questions. Some of them will be about job security. "Am I next?" is the unspoken question in the room after any termination, even when the circumstances were entirely different from anyone else's situation. Your 1:1s in the week after a firing are the place to address this directly, not defensively, and with whatever honest reassurance you can actually offer.

What you don't do: badmouth the person who left. Not in private, not to the team, not to close allies on the team. This is partly decency. It's also partly pragmatic — the stories managers tell about the people they've fired have a way of circling back, and they reveal more about the manager than the employee.

If the person had relationships inside the team — and they usually did — those team members are allowed to feel complicated things. Grief, anger, relief, guilt. All of those are legitimate. You don't need to manage the emotions. You just need to not punish them.

The Dignity Question

The chapter title names something that's easy to lose in the mechanics of decision frameworks and legal checklists. Dignity. It's the part that doesn't show up in the HR training.

Fournier writes, in The Manager's Path, that the hardest situations are the ones where someone is not bad, not malicious, just not right for the role. Those are the majority of terminations. Not the fraud case, not the hostile conduct situation — those are comparatively simple. The person who was a strong mid-level engineer but couldn't grow into the staff role you needed. The marketing manager who built great campaigns but fell apart when the team scaled. The person whose skills were perfect for who you were two years ago and simply aren't anymore.

Those firings hurt differently. They require you to hold two truths: this person is capable and decent and genuinely tried, and this is still the right call.

Preserving dignity in that moment looks like this: being honest about the reason without being brutal about it, allowing the person to leave without a walk of shame if the role allows for it, following up on any commitments you make (a reference call, an introduction to a contact, a fair description of what they accomplished), and remembering that this is probably one of the worst days of their professional life. Not because you're a bad person. Because it just is.

The manager who confuses dignity-preserving with the avoidance behaviors we talked about earlier — the softened language, the extended false hope, the PIP you don't believe in — hasn't found a more humane path. They've found a longer path to the same outcome with more accumulated damage along the way.

Being clear is kind. It's also harder. Those two things are almost always true at the same time in management.

A Note on What This Does to You

The first person you fire stays with you. I don't care how certain you are that it was the right decision — and it might have been unambiguously the right decision — there's still a weight to it. You're not supposed to be breezy. The managers who are breezy about letting people go are not the ones you want to be.

But there's a difference between the appropriate gravity of the thing and a guilt that paralyzes you from doing it when it's necessary. Grove was direct about this in High Output Management: the manager's job is to maximize the output of the organization, and that sometimes means making calls that no one likes, including

you. That's not cruelty. It's the actual job.

You will do this wrong at least once. Most managers do. The wrong kind of PIP, the conversation that came too late, the framing that felt careful and landed as hostile. You will learn from that the way you learn from most management failures: by sitting with it, understanding what you'd do differently, and not confusing the lesson for a reason to avoid the next hard decision.

The goal is not to become someone for whom firing is easy. The goal is to become someone for whom it's honest.

That's different, and it's enough.

CHAPTER VIII

The Performance Review, Actually Useful Version

Every December, or March, or whenever your company's HR calendar decrees, a collective groan moves through the organization like a weather system. Managers open blank performance review templates. Direct reports brace themselves. Someone in HR sends a reminder email with a cheerful subject line — "Time for our Annual Review Cycle! Ø<ß " — that nobody finds cheerful. And then everyone writes documents that will be skimmed, filed, and forgotten by February.

This is not cynicism. This is a description of most performance review processes operating exactly as designed.

The design is the problem.

Most review templates are built to satisfy legal and HR requirements, which is a legitimate goal, but a completely different goal from actually developing your people. When you optimize a document for defensibility, you get defensible documents. When you optimize for development, you get something that can actually change how a person works. Almost no company does the second thing. Which means almost no manager has been shown how, and almost no first-time manager comes in with any model other than "write something that won't get me in trouble."

Andy Grove wrote, in High Output Management, that "the output of the review is the manager's objective assessment of the subordinate's performance, plus the future action plan." Future. The word is doing real work there. Grove's view — and he ran Intel, so he'd seen a few thousand performance conversations — was that the entire purpose of the review is to change what happens next, not to document what already happened. The past is evidence. It is not the point.

Once you accept that, the whole exercise looks different. You're not writing a verdict. You're writing a development brief.

What Specific Actually Means

Here is a sentence from a real performance review. I've changed the name.

"Marcus consistently demonstrates strong communication skills and has been a valuable contributor to the team."

That sentence tells Marcus nothing. It tells his future manager nothing. It tells the HR system that Marcus didn't do anything fireable. That's it. And it took maybe forty-five seconds to write, which is probably about how long it was given.

Contrast that with this: "Marcus rewrote the onboarding documentation for the payments API in Q3. Before the rewrite, new engineers reported an average of 2.3 hours to set up a working local environment; after, that dropped to under 40 minutes. Three separate engineers on other teams flagged this to me unprompted."

Same Marcus. Totally different sentence. The second version is specific enough that Marcus can repeat the behavior. It's specific enough that his next manager understands what he actually does. It's specific enough that it means something in a calibration meeting with seven other managers who all think their people are "strong contributors."

Specificity has three components. You need the behavior — what Marcus actually did, not a personality trait. You need the impact — what changed because of that behavior, in numbers or concrete outcomes where you can get them. And you need the signal — how you know this, whether it came from Marcus directly, from peers, from data, from your own observation. Signal matters because it tells the reader how much confidence to place in the assessment.

Julie Zhuo's framing in The Making of a Manager is useful here: describe the impact as if you're explaining it to someone who doesn't know the person. If you can read the sentence cold, with no prior knowledge, and understand what the person did and why it mattered — you've written a useful review. If you need context to decode it, rewrite it.

Run every paragraph through that test. Most first drafts fail it badly.

The Before-and-After

In my second year managing, I had a senior engineer on my team — I'll call her Priya. Priya was genuinely excellent. Also occasionally a nightmare to work with, particularly when she disagreed with technical decisions she hadn't been part of making. She'd go quiet in meetings, then relitigate everything on Slack afterward with whoever would listen. It created drag. It slowed decisions down. And I'd been avoiding putting it clearly in writing because I liked Priya and I told myself the feedback was "ongoing."

Here is an excerpt from what I wrote in her first annual review under me.

"Priya is a technically exceptional engineer who brings deep expertise to complex problems. At times, her communication style can create friction, and there's opportunity to grow in how she engages with cross-functional stakeholders."

Read that again slowly. I said nothing. "Communication style can create friction" could mean she sends emails too long. "Opportunity to grow in how she engages with cross-functional stakeholders" — what does that even mean? I wrote it because it felt honest enough to be defensible and vague enough not to require a difficult conversation. Which is exactly the wrong set of priorities.

Here's what I rewrote that section to say, after the review cycle had closed and I was doing the post-mortem on my own writing.

"Priya's technical contributions in this period were significant — she led the architecture review for the data pipeline migration and her work directly reduced our error rate from 4.2% to under 0.8% over six weeks. The area where she has the most room to grow is in how she engages with decisions she wasn't part of making. In three separate incidents this year (the API versioning decision in March, the deployment cadence change in July, and the team tooling switch in October), Priya expressed disagreement after decisions were finalized rather than during the discussion window, and in two of those cases, she continued discussing the decision in Slack channels in ways that slowed the team's ability to move forward. The goal for the next review period is for Priya to practice stating objections during the designated review window, and to disengage from relitigating closed decisions — which she and I have discussed directly and which she has agreed is the behavior to change."

Night and day. The second version is harder to write. It requires you to name specific incidents, which means you have to have been paying attention all year, not just the last three weeks before the template is due. It requires you to have already had the conversation with Priya, which means you can't use the review as a surprise ambush. And it requires you to say something uncomfortable on the record, which means accepting the discomfort.

But it's also the only version that could actually change anything. Priya reads the first version and nods and nothing changes, because nothing was asked of her. Priya reads the second version and knows exactly what the target is.

She also, to her credit, improved on it in the next cycle. Because there was something specific to improve on.

The Architecture of a Useful Review

You don't need a long review. You need a precise one. Three to five areas of substance, each written with the specificity rule applied, beats twelve paragraphs of generalities every time. Here's the structure I've settled on.

Contributions and impact. Two or three specific examples of what the person did and what it produced. Numbers where you can get them. Named projects. Dates if they help. This section is backward-looking, but it's evidence you'll use to argue for ratings and compensation, so it needs to be airtight.

How they work. This is where you address the invisible stuff — how the person shows up in collaboration, how they communicate, whether they pull others up or create drag, how they handle ambiguity. This is also where most vague review language lives, so be ruthless with yourself. "Good communicator" needs to become "consistently writes project specs that other engineers can act on without follow-up questions," or it needs to come out.

Growth areas. Not weaknesses. Not flaws. Areas where growing would create the most leverage for this person in the next twelve months. One or two is plenty. More than two and the message gets diffuse. Pick the one or two that matter most for where this person wants to go, and make them specific enough that you could both describe measurable progress at the next annual check-in.

Forward-looking goals. Grove's north star lives here. This section should answer: what would a strong performance look like next year, and what's the support structure to get there? Not aspirational boilerplate. Actual goals with enough specificity that you'll both know in December whether you hit them.

Calibration: The Part Nobody Tells You About

Writing good reviews is one problem. The second, harder problem is making sure your ratings mean something across your team and across your peers' teams.

Calibration sessions — the meetings where managers compare their assessments and ratings before they're finalized — are where well-written reviews pay off and where vague reviews expose you. If you wrote "Marcus is a strong contributor," you will be asked to justify why Marcus is a "Meets Expectations" versus a "Exceeds Expectations," and "strong contributor" won't hold up. If you wrote the three-sentence version with numbers and peer feedback, you can make that case.

Calibration also exists because of a predictable human tendency: most managers rate their direct reports higher than average, which is a mathematical impossibility when "average" is defined by the same population. Research by industrial psychologists going back decades — and replicated consistently in corporate settings — shows that when managers rate independently, distributions tend to cluster heavily at the top end. Everyone thinks their team is exceptional. Statistically, that's not possible.

The calibration session exists to correct for this. Which means your job, going in, is to be able to distinguish between your own genuine high performers and the people you've been generous with because you like them, or because giving someone a middling rating feels bad, or because you haven't had the courage to give them feedback clearly enough that you can now put it in writing.

Camille Fournier, in The Manager's Path, makes this point sharply: a manager who gives inflated ratings isn't being kind — they're being cowardly, and the cost gets paid by the team. When everyone is "Exceeds Expectations," the label means nothing, which means your actual high performers get no signal that they're genuinely exceptional, and your lower performers get no signal that they need to lift their game. You've made the whole system useless in the name of avoiding a few uncomfortable conversations.

Do not inflate. Your high performers are watching whether their exceptional work is distinguishable from adequate work. If it isn't, they will find an organization where it is.

Go into calibration with your assessments written out and your evidence ready. Expect to be challenged. The challenge is the point. A good calibration session should feel slightly uncomfortable — you'll defend some ratings, you'll revise others, and you'll come out with a distribution that reflects the actual spread of your team's performance rather than your generosity.

One practical note: take notes during calibration on how other managers describe their high performers. You'll learn a lot about the standards the organization is actually applying, which helps you calibrate your own thinking for the next cycle.

Delivering the Review Conversation

The document matters. The conversation matters more.

Most managers make two mistakes in the actual review meeting. The first is turning it into a reading exercise — walking the direct report through the document paragraph by paragraph, narrating their own writing. This is excruciating for both people and accomplishes almost nothing. The direct report should have read the document before the meeting. If they haven't, give them five minutes to read it now. Don't read it aloud to them like a bedtime story.

The second mistake is treating the conversation as a monologue. You talk, they receive. This is not a development conversation. This is a transmission. And transmission doesn't change behavior — engagement does.

Structure the conversation to flip the ratio. Open by asking the person to share their own assessment first. This does two things: it tells you how well-calibrated they are (critical information), and it makes them feel heard before they hear from you, which makes them meaningfully more receptive to whatever comes next. Ask something like, "Before we get into what I wrote, I'd love to hear where you think you had the most impact this year, and where you think you have the most room to grow."

Then listen. Actually listen, not just wait for them to finish so you can start.

When you share your written assessment, lead with contributions — what they did and why it mattered. Give that real airtime, not a perfunctory thirty seconds before you get to "areas for growth." People need to feel that their work has been genuinely seen before they can hear anything that challenges them.

When you do reach growth areas, present them as forward-looking investments, not report card grades. Grove's framing is useful here again: this is about future performance. "Here's what I think would create the most leverage for you in the next twelve months" is a different frame than "here's what you did wrong this year." The outcome you want from this conversation is a person who walks out feeling like they have a clearer picture of where to put their energy — not a person who walks out feeling judged and slightly humiliated.

Ask for their reaction after each major point. "Does this match how you see it?" is a simple question and a powerful one. You'll learn quickly whether you're aligned or whether there's a gap to work through. Gaps aren't bad — they're information. If someone thinks they're performing at a level you assessed as significantly lower, that gap itself needs a conversation. You can't address it if you don't surface it.

End the meeting by confirming the forward-looking goals together. Not your goals for them. Goals you've built together in the conversation. There's a real difference in ownership, and it shows up in what happens next.

The whole meeting, for most people, runs forty-five minutes to an hour. For high performers with no surprises, sometimes thirty minutes. For complex situations — a performance issue, a rating they'll contest, a

person who's been struggling — plan for ninety. Don't do these back-to-back if you can avoid it. You need bandwidth between conversations to reset.

The Thing About Surprises

Nothing should be a surprise in a performance review. Not the rating. Not the growth areas. Not the specific incidents you're citing.

If you've been running your 1:1s well — giving feedback in the moment, naming problems when they happen, not saving everything for December — the review should feel like a summary, not a revelation. The direct report should be able to predict roughly what you'll say, because you've been saying versions of it all year.

If your review is full of surprises, that's not a documentation problem. That's a feedback problem, and the review is where it catches up with you.

Camille Fournier calls this "no surprises" principle one of the baseline expectations of a fair management relationship. The person being reviewed has the right to know, in real time, what they're doing well and where they're falling short. Saving it up — because it's uncomfortable to say now, because you're hoping it'll resolve itself, because you just haven't gotten around to it — is a failure of the ongoing job, not a reasonable strategy for the annual one.

The corollary to this: if someone is genuinely on a path toward a performance improvement plan or a role change, they need to know that well before review season. Nobody should learn they're in serious standing trouble from an annual review document. That's not a review. That's an ambush.

A Note on Ratings

Different companies use different rating systems — four points, five points, narrative-only, stacked ranking (please don't), objectives-and-key-results-based. Whatever your company uses, a few principles hold.

Be honest about what the middle means. At most companies, the middle rating ("Meets Expectations" or its equivalent) describes the majority of a healthy team. A person who meets expectations is doing their job well. That's not a backhanded compliment. That's a description of solid performance. Treat it that way in your conversation. "Meets Expectations" delivered apologetically, as if you're breaking bad news, teaches people that the middle is something to be ashamed of. It isn't. It's where most good work lives.

Be honest about what the top means. At most companies, a "Consistently Exceeds" or equivalent rating should be genuinely rare — maybe 10 to 20 percent of your team in a healthy cycle, lower in a calibrated organization. If you're giving your highest rating to 60 percent of your people, either you have an extraordinary team (possible, but check your reasoning) or you've inflated the standard into meaninglessness.

And be honest about the bottom. A "Below Expectations" rating, given fairly and with specificity, is not a punishment — it's a signal. A person who doesn't know they're underperforming can't course-correct. A

person who does know has a fighting chance.

The hardest rating conversations are with people who are excellent at parts of their job but struggling significantly in others. There's no clean answer here — you use judgment, you document both dimensions clearly, and you make sure the rating reflects the net assessment you'd give if you were explaining this to their next manager. That's the test. What would you want their next manager to know?

The Year-Round Investment

Here's the thing about good performance reviews: the writing happens in December (or March, or whenever), but the material is collected all year.

The managers who write the clearest, most specific reviews are not necessarily the most talented writers. They're the ones who keep a running document — a note, a folder, a Notion page, whatever format survives contact with your actual habits — where they drop specific observations throughout the year. A quick note after a meeting where someone handled a difficult client exceptionally well. A note when someone missed a deadline and you talked about it. A note about the piece of work that shipped and performed beyond expectations.

This takes maybe three minutes after any notable event. Aggregated over twelve months, it gives you a year's worth of specific evidence to draw from. Without it, you're relying on memory, which is systematically biased toward the last sixty days and toward events with emotional salience — which means recent successes and recent failures get overweighted, and the quiet steady work that defines most people's performance gets underweighted.

Julie Zhuo describes this as the "snapshots vs. movie" problem: a review based on memory gives you a few snapshots; a review based on ongoing notes gives you something closer to the full film. The person who had a rough Q1 but an exceptional Q3 and Q4 deserves an assessment that reflects all four quarters, not just the ones you remember most clearly.

Set a calendar reminder — "update performance notes" — for the last Friday of every month. It takes less time than you think. It transforms what you're able to write in review season.

What Grove Got Right

Andy Grove didn't have much patience for reviews as ritual. His position, stated plainly in High Output Management, was that a manager who doesn't take the review seriously is failing in one of their most basic obligations. He estimated that preparing a good review for a single subordinate should take three to four hours — not counting the notes you've taken all year. Three to four hours of actual work, not template-filling.

Most managers, if they're honest, spend closer to forty-five minutes.

The math on why the longer investment is worth it isn't complicated. You might manage eight people. A genuinely useful review for each of those eight people — one that gives them clear direction for the next

twelve months — compounds over 365 days of better-targeted work. A vague review compounds differently: the person is slightly less clear on what matters, slightly less motivated by the sense that their work is seen, slightly less likely to repeat the right behaviors or change the wrong ones. Multiply that across eight people for a year.

Three extra hours per person is not an expensive investment against that math.

The review isn't paperwork. It's a set of instructions for how you want the next year to go, written in a language the person on your team can actually act on.

Write it that way.

One final thing. After I rewrote Priya's review — the version I never sent, because the cycle had already closed — I made a copy, changed the name to mine, and read it as if I were my own manager writing about me. It was an uncomfortable exercise. The growth areas landed differently when I was on the receiving end of them.

That's the point. Before you finalize any review, read it once with the other person in your head. Not to soften it — to test whether it's true, specific, and useful. Whether you'd want to receive it. Whether you'd walk out of the conversation knowing what to do next.

If the answer is yes, send it.

CHAPTER IX

Career Conversations: Not Just the Annual Check-In

The thing nobody tells you about losing your best people is how obvious it looks in retrospect.

The signals were there — the slightly flatter energy in the last few 1:1s, the stretch project she stopped bringing up after you kept punting it to next quarter, the LinkedIn profile update you caught by accident because the algorithm surfaced it on a Tuesday morning. Then she hands in her notice and you do the exit interview, and she says, with genuine kindness, "I just needed more room to grow." And you nod, and she's right, and somewhere in the back of your brain you're already replaying six months of conversations where you never once asked what growing meant to her.

This is the failure mode. Not cruelty. Not incompetence exactly. Just a manager who treated career development as a December ritual — a box to check, a form to file — and a person who eventually stopped believing anyone was paying attention.

Her next job? She got promoted into it eight months later. It was a role your company had, and could have structured for her, if anyone had mapped her ambitions against what the organization actually needed.

That stings. It should.

Why the Annual Review Doesn't Do This Job

The performance review is the wrong tool for career development, even when it's done well. A good review — and chapter 8 covers what that looks like — is a structured look backward. It asks: what did you accomplish, how did you operate, where do you need to grow? Those are fair questions. They just aren't the same as: where do you want to be in three years, and what do we need to do between now and then?

Mixing the two conversations is like asking someone to reflect on their driving record and plan their road trip at the same time. One requires honest reckoning, the other requires imagination. The emotional registers are completely different. The backward-looking conversation involves accountability; the forward-looking one requires trust. When you bolt them together once a year, both suffer.

Camille Fournier addresses this directly in The Manager's Path. Her framing is clean and worth holding onto: the manager's job is to create opportunity, the employee's job is to seize it. That division of labor sounds simple until you realize what it actually demands of you. Creating opportunity isn't a passive act. You don't create opportunity by waiting for the org chart to rearrange itself or hoping a stretch project falls from the sky. You create it by knowing — specifically and continuously — what each person on your team wants to build toward, and then engineering the conditions where that becomes possible.

That requires talking to people. Regularly. Before the annual review. Before the exit interview.

The Lightweight Cadence (and Why It Stays Lightweight)

When managers hear "ongoing career conversations," they picture a new recurring meeting, a new agenda template, a new thing to be bad at. That's not what this is. The cadence I'm describing doesn't add a single calendar event if you run 1:1s already. It adds intention to conversations you're already having.

Here's the rough structure that works:

Every 1:1, informally: One question, once in a while, about what the person is excited about or what they're finding hard. Not a formal development check-in — just keeping the ambient signal alive. "What's been energizing you this week?" gets you real information about where someone is engaged and where they're going through motions.

Every six weeks or so: A slightly deeper beat. This is where you check in on the longer arc. Is the project they're leading actually teaching them what they hoped it would? Is there a skill gap showing up that you could address? Five minutes at the end of a 1:1, consistently, beats a formal quarterly meeting that never actually happens.

Twice a year: A dedicated career conversation, forty-five minutes, no status updates, explicitly focused on the longer view. What do they want to be doing in two or three years? What's their honest read on where they are versus where they want to be? What would they do if they weren't worried about whether you'd approve?

That's it. No framework to buy. No software to implement. Two longer conversations a year, supported by a low-level current running through your regular 1:1s. The reason it stays lightweight is that the value isn't in the formality — it's in the consistency. Your team needs to believe that you think about their futures between January and December. The cadence creates that belief.

The First Conversation: Asking About Long-Range Ambitions

The hardest career conversation is the first one, because neither of you knows exactly how honest to be yet.

Most people, when asked "where do you see yourself in five years?", immediately perform an answer. They give you the professional, manager-appropriate version — something ambitious but not threatening, something aligned with the company's interests, something that won't make them sound disloyal or impractical. What they don't do is tell you the truth.

The truth, for a lot of people, is messier. Some of your reports genuinely don't know what they want beyond a general sense of "more" — more challenge, more money, more autonomy. Some know exactly what they want but have decided it's not safe to say it around their manager. Some want something that has nothing to do with their current team, and they've been quietly building toward it on their own because it never occurred to them that you might help.

Your job in this conversation is to make the real answer feel safe enough to say. A few things that help:

Go first, a little. Share something honest about your own career trajectory — ideally something that wasn't a straight line. When I told one of my reports that I'd spent eighteen months in a role I thought I wanted before realizing it was completely wrong for how I'm wired, her entire posture changed. She stopped performing. She started talking.

Ask better questions. "Where do you see yourself in five years?" is a garbage question — it's been asked in so many performative contexts that it triggers a canned response. Try instead: "What kind of work makes you feel like you're operating at your best?" Or: "Is there something you were hoping this role would teach you that it hasn't yet?" Or the one I like best, borrowed loosely from Fournier's approach: "If you could design your ideal role for two years from now — no constraints — what does it look like?" The constraint removal is important. You want to hear what they actually want, not what they've decided is realistic given the team you're running.

Don't react, not immediately. When someone tells you something that surprises you — "I want to move into product management" or "I'm thinking about starting a company in the next five years" — your face is doing something. Watch it. The instinct is to immediately problem-solve or to register concern. Both close the conversation. Just ask another question. "What draws you toward that?" You can process your own reaction later.

Take notes you share. After the conversation, write a brief summary of what you heard and send it to them. This does two things: it demonstrates that you were actually listening, and it creates a reference point you can both return to. "Back in February you mentioned you wanted to build more cross-functional experience — how's that been going?" lands completely differently than asking someone to repeat themselves every six months because you forgot what they said.

Mapping Projects to Skill Gaps

Good intentions about career development don't do much without a mechanism for acting on them. The mechanism is simple: you know what your team members want to get better at, and you know what work needs to get done on your team. Your job is to make those things overlap as often as possible.

This sounds obvious. It's harder than it sounds.

Start by listing the skills each person has identified as important to their growth. Not skills you think they need — skills they've told you they want to develop. These might include things like running a cross-functional project, presenting to senior leadership, making architectural decisions, mentoring junior colleagues, building client relationships. Whatever came out of the career conversation.

Then look at your next quarter's worth of work. Every team has projects that require different mixes of skills. Some projects have high visibility; some are technically complex; some require managing stakeholders outside the team; some are straightforward enough to be owned completely by someone who's never done it

before. Match intentionally. When you have a project that requires coordinating with three other teams, give it to the person who said they want to practice cross-functional influence, not necessarily to the person who's already good at it.

There are two failure modes here worth naming. The first is always giving the important work to the people who are already proven — which feels safe but doesn't develop anyone and eventually bores your strongest performers. The second is handing someone a stretch so large they drown. The difference between a development opportunity and a setup-to-fail isn't the size of the challenge; it's the scaffolding around it. You can assign someone a project that's genuinely hard for them as long as you stay close enough to catch them before they go too far wrong and debrief the experience afterward.

Andy Grove's framework in High Output Management is useful here. Grove argues that a manager's output is the output of the team they run. If that's true — and I think it is — then the fastest way to increase your team's output over a multi-year horizon isn't to hire more people or squeeze more hours out of the ones you have. It's to make each person on your team more capable, faster. Development isn't a nice-to-have that you do when things are quiet. It's a core productivity lever. The manager who treats it that way compounds their team's capability every quarter. The manager who treats it as a once-a-year HR requirement doesn't.

I had a senior engineer on my team — I'll call him Daniel — who kept telling me he wanted to move into a technical lead role. He was technically excellent. The gap was influence: he got impatient in design reviews and had a habit of solving problems before other people had finished describing them. Not a fatal flaw, just a thing that needed work. The wrong move would have been to wait for a "technical lead opportunity" to appear and then hand him the role and hope for the best. Instead, I gave him increasing ownership of our system design review process — weekly, low-stakes, real decisions — and we debriefed after every session. What landed well. Where he'd steamrolled someone accidentally. Six months later he ran his first large cross-team design review and it went well enough that two engineers from a different team mentioned it to their manager. His own analysis of why it went better than his earlier attempts was more precise than anything I could have told him. He'd built the skill. Not because I handed him a promotion, but because I gave him a place to practice with enough feedback to adjust.

That's what Fournier means by creating opportunity. Not fabricating it out of thin air — engineering it out of the real work in front of you.

When the Path Isn't on Your Team

Here's the conversation most managers never have, and it's the one that actually builds long-term trust.

Sometimes the thing your direct report wants isn't available where you are. The person who wants to be a VP of Engineering in four years needs experience you can't give them in a twenty-person startup without a VP track. The person who wants to pivot into machine learning is going to hit a ceiling on a team that does mostly backend web services. The person who told you — honestly, because you created the conditions for honesty — that they're thinking about starting their own company in three years needs something from their

career that no single employer can provide.

When that's the case, you have two options.

Option one: say nothing, or say something vague and reassuring, and hope the conversation doesn't come up again. This is by far the more popular choice. It also means that when that person eventually leaves — and they will — you've spent whatever time they had on your team pretending the gap didn't exist. You've wasted their career planning time and your relationship capital.

Option two: be honest. Tell them what you actually see. Tell them where the ceiling is on your team and what, specifically, it means for what they're trying to build toward. And then — this is the part that takes some courage — help them think through what to do about it.

That might mean advocating for a lateral move to a team that has more of what they need. It might mean helping them find a mentor outside your team who has navigated a similar path. It might mean making an introduction to someone at a company where the role they want actually exists. It might mean having an explicit conversation about a timeline — "I don't have a technical lead role available now, but I think I could build the case for one in the next nine to twelve months if we work toward it together."

None of these conversations are comfortable. The first time I had an honest version of one — telling an engineer that if she wanted to get into venture capital, my team wasn't going to be the main vehicle for that, and here's what I thought might actually help — I fully expected her to put in her notice the next week. She didn't. She stayed for another fourteen months, did excellent work, and left for a business school program with a reference letter I was genuinely happy to write. More importantly, in those fourteen months she trusted me in a way she hadn't before the conversation. Because I had chosen her actual interests over my short-term staffing convenience.

Julie Zhuo writes about this tension in The Making of a Manager — the manager who conflates team stability with employee development, who protects their roster at the cost of the people in it. The irony is that the managers who hold most tightly to their people drive them away faster. The managers who engage honestly with what people want, even when that creates awkward conversations, tend to retain people longer. Because trust is a retention mechanism. People don't leave managers who are honest with them about hard things — or at least, they don't leave without a much longer conversation first.

The "I Don't Know What I Want" Problem

Not every person on your team is going to arrive at a career conversation with a five-year plan. Some people are genuinely uncertain. Some are early enough in their careers that the questions feel abstract. Some have been in survival mode — difficult project, difficult personal circumstances — and haven't had the mental bandwidth to think beyond the next sprint.

That's fine. Honest uncertainty is infinitely more useful to you than a polished answer that doesn't reflect what someone actually thinks.

For the person who doesn't know what they want, the career conversation shifts slightly. Instead of asking about long-range ambitions, you're helping them build the self-awareness to eventually have ambitions. Questions that help: "What has been the best day at work in the last six months — what made it good?" "Is there anyone whose role you've looked at and thought, I'd find that interesting?" "What's something you've always been curious about professionally but never made time for?"

You're not trying to install a plan into someone who doesn't have one. You're trying to help them listen to themselves. That takes more than one conversation, but it's still a conversation worth having.

The other thing worth saying about uncertainty: sometimes it's a signal that something is wrong. An engineer who was enthusiastic and goal-directed twelve months ago and is now shrugging at every question about the future might not be uncertain — they might be disengaged. Or they might be quietly considering a change they haven't mentioned to you yet. "I'm not sure" can mean "I genuinely haven't thought about it" or it can mean "I don't trust that my real answer is safe here." You'll learn to tell the difference, mostly by paying attention over time. The shape of someone's disengagement is different from the shape of their uncertainty.

Closing the Loop: What Happens After the Conversation

The career conversation that doesn't lead anywhere is worse than no career conversation at all. It raises expectations, burns trust, and confirms the suspicion that this was performative. So after every significant career discussion, you need a clear next step — even if it's a small one.

Small is fine. Small is often exactly right. "I'm going to introduce you to the head of data science by the end of this month" is a next step. "You said you want to improve your public speaking — let's put you on the agenda for next quarter's all-hands to present the results of the migration project" is a next step. "I'm going to flag your interest in the senior role to my boss when we have our next calibration conversation" is a next step. These don't require budget or headcount or organizational restructuring. They require you to do the thing you said you'd do, and then to come back to the topic.

The following 1:1 should reference what you discussed. Not a full formal review — just a quick acknowledgment that the conversation is alive. "I made that intro — did you two connect?" or "How are you feeling about the presentation coming up?" That's it. That's how a conversation becomes a cadence instead of an isolated event.

Fournier is blunt about the stakes: the managers who do this consistently are the ones people want to work for. Not because they're the most technically brilliant or the most senior or the best resourced. Because they make their direct reports feel like someone is actually thinking about their futures. In a world where every company claims to care about employee development and most of them are lying through their org charts, a manager who actually shows up — consistently, with memory, with follow-through — is genuinely unusual. That's your competitive advantage as a manager if you want one.

Before your next round of 1:1s this week, ask yourself what you actually know about what each person on your team wants their career to look like in two or three years. Not what they're good at. Not what project they're on. What they want.

If the honest answer is "not much," that's your starting point. Not a failure — a starting point. Schedule the conversation. Prepare a couple of good questions. And then, when they answer, don't rush to fix anything or promise anything. Just listen. Take notes you actually keep. Come back to it in six weeks.

That's it. That's the whole thing. Not a system, not a program. A conversation, and then another one, and then another one after that — until the person across the table from you knows, in their bones, that someone is paying attention.

That's what makes people stay. And when they eventually go — because they will, some of them — it's what makes them remember you as someone who gave a damn.

CHAPTER X

Managing Up: Your Boss Is Also Your User

The phrase "managing up" has a reputation problem. Say it in a room full of new managers and at least two of them will make a face — the face people make when someone suggests something they find mildly distasteful, like expensing a team lunch that's clearly just personal dining. They've filed managing up next to "playing politics" and "optics" in the mental drawer labeled things I refuse to do because I have integrity.

That instinct is understandable. Also wrong.

Managing up isn't about flattering your boss or curating your visibility or engineering the perception that you're doing more than you are. It's something far more practical: making your manager's job easier by giving them the right information, at the right time, at the right level of detail, so they can do their job without constantly excavating yours. Think of it as a service design problem. Your boss is a user. You are building the interface through which they experience your team's work. A bad interface — sporadic updates, wrong granularity, problems that surface only after they've metastasized — creates friction, erodes trust, and eventually makes you a liability they have to manage around instead of a leader they can lean on.

Andy Grove put it plainly in High Output Management: your manager's output is only as good as the inputs they receive. They're not omniscient. They're managing their own portfolio of problems, probably three layers of organizational complexity, and a calendar that would make a normal person weep. Your job is to be the one person on their team who makes them feel, at least briefly, like things are under control.

That's managing up. It's professional. It's unglamorous. And almost nobody does it well.

The Skip-Level That Blindsided Everyone

Let me tell you about the worst professional meeting I sat through in my first year as a manager — and the part that stings is that I caused it.

It was month four. I had inherited a team of six engineers working on our data pipeline, and I had spent those four months doing what I thought was managing: shipping features, running 1:1s, handling team drama, keeping the velocity numbers green. I was doing the job. I had evidence. I felt good about myself.

My manager, Clara, had a quarterly ritual with her boss — call him Derek — where they reviewed each team's health, roadmap confidence, and any significant risks. Clara had been doing this sync for two years. It was routine.

What I didn't know was that Derek had started asking team leads to join briefly for their segment of the review. Clara had mentioned it in passing once, weeks earlier, and I had filed it under "noted" without actually registering it as something that required preparation. So when I sat down across from Derek for

approximately eleven minutes, he asked me three questions: What's the biggest technical risk in the pipeline right now? What do you need from leadership that you're not getting? And how confident are you that the Q3 roadmap is achievable?

I knew the answers to all three. I had been living inside these problems. But Clara had not heard any of them — not in the form Derek was now hearing them for the first time, out of my mouth, in real time, in front of him. The technical risk I named — a dependency on a third-party data vendor whose SLA had been quietly degrading for six weeks — was news to her. The resource gap I described was news to her. The quiet concern I had about Q3 was news to her.

She kept a professionally neutral expression throughout. She is very good at her job. But I could see, in the particular stillness of her face, that she was logging everything I said and simultaneously calculating how she had not known any of it.

After the meeting she found me in the hallway. She did not raise her voice. She said, quietly: "I looked uninformed in front of my boss. In four months, you have not given me one status update about the vendor SLA situation. Why not?"

The honest answer was that I had been managing the problem myself and I figured as long as I had it handled, there was nothing to surface. I had confused working on a problem with having resolved the problem. I had been optimizing for looking capable — self-sufficient, on top of things — rather than being transparent with the person who needed to know.

It cost us both trust that took another three months to rebuild.

Information at the Right Altitude

Here's the thing about the vendor SLA situation: Clara did not need a daily update on it. She did not need the detailed technical breakdown of why the third-party latency was spiking or the three mitigation approaches my team was evaluating. That would have been noise. What she needed was the signal: there is a risk to Q3, it is being actively managed, and here is the threshold at which I will need your help.

One sentence. Maybe two.

The failure wasn't that I hadn't sent her a seventeen-page brief. The failure was that I hadn't calibrated what altitude she needed to operate at, and then given her information at that altitude. This is actually a learnable skill, and the best way to build it is to ask directly.

In your next one-on-one with your manager — and you should be having these regularly, just as your direct reports have them with you — ask two questions. First: what's the level of detail that's actually useful to you? Second: what would make you feel blindsided, and how do I make sure that never happens? Most managers will tell you exactly what they want if you ask plainly. They'll say something like "I don't need the day-to-day, I need to know when something is off-track before it's too late to fix it" or "I like a short written summary on Fridays, I don't love being ambushed with status in meetings."

Write down what they say. Hold yourself to it.

Julie Zhuo, in The Making of a Manager, describes the relationship between a manager and their own manager as something that requires active maintenance — not just showing up and hoping for the best. "Managing up," she writes, "means understanding what your manager cares about, then helping them achieve it." That sounds obvious. It is almost universally practiced badly.

What your manager cares about is rarely the same as what you care about. You care about the quality of the code review process. They care about whether the roadmap is on track and whether your team is a net positive or a net drain on the organization's overall momentum. You care about the nuances of a performance situation with one engineer. They care about whether it's going to require HR involvement or quietly resolve itself. The information your boss needs is a compressed, executive-summary version of the information you're living inside every day. Translating between those two levels — that's the skill.

A practical framework: think of your updates as a three-tier system. Green means things are proceeding as expected and you have no decisions that require their input. You can batch green updates into a short weekly note or mention them briefly in your 1:1. Yellow means something has changed from the baseline and they should know about it, even if you're handling it, because it might affect their plans or their own conversations upward. Yellow things belong in your next 1:1, or in a brief Slack message if it's time-sensitive. Red means a problem has crossed a threshold where you either need their help, or they will look uninformed if they hear about it from someone else first. Red things get surfaced immediately. Not "next week," not "at the all-hands." Now.

The vendor SLA situation was a yellow that had been turning slowly red for six weeks. I never surfaced it. That's the mistake.

Surface Problems Before They're Fully Formed

There's a version of professionalism that says: don't bring your manager a problem unless you also bring a solution. Don't come empty-handed. Don't be the person who just drops a crisis in someone's lap.

This is good advice in the wrong hands. Taken too literally, it becomes a reason to say nothing until you've figured everything out — which means your manager finds out about problems only after they've been festering long enough for you to solve them, or (worse) long enough that they can no longer be solved.

Camille Fournier addresses this in The Manager's Path when she talks about the manager's obligation to make information flow upward clearly: "Part of your job is to know when to escalate." That seems simple. In practice, most new managers delay escalation because they're afraid it signals incompetence. They want to be the person who handled it. The irony is that handling something quietly and having it surface later, in a skip-level meeting or a hallway conversation or an email you weren't copied on, signals exactly the incompetence they were trying to conceal.

The reframe is this: surfacing a problem early is not the same as being unable to handle it. It's giving your manager the ability to help you if you need it, prepare their own communication if they need to, and avoid being blindsided. You can surface a problem and simultaneously communicate your plan for addressing it. "I wanted you to know that X is happening. Here's what I'm doing about it. I'll let you know if it develops beyond what I can handle." That's not weakness. That's the job.

A useful rule of thumb: if you would be embarrassed for your boss to hear about something from someone other than you, that's the thing you should be telling them. Today.

Pushing Back Without Burning the Bridge

Not every direction you receive from your manager will be correct. Some of it will be based on incomplete information. Some of it will reflect a priority trade-off that makes sense at their level but doesn't account for something only you can see. Some of it will just be wrong.

Knowing when and how to push back is one of the hardest skills in the first-time manager toolkit, because the cost of doing it badly is high — you can come across as insubordinate, or obstructionist, or simply immature. And so most new managers overcorrect: they swallow their disagreement, execute direction they think is flawed, and wait to say "I told you so" when things go sideways. This is also wrong, just in the opposite direction.

Grove's operating principle was that decisions should be made at the level closest to where the information exists. If your manager is making a call based on information that you have and they don't — customer context, engineering constraints, team capacity, interpersonal history — then letting them make that call without surfacing what you know is a failure of your job, not deference. You are not being professionally humble by staying quiet. You are being professionally negligent.

The mechanics of a good pushback:

Timing matters more than most people realize. The right moment to push back on a direction is before the decision is fully locked, not after. Once a plan is announced to the broader team or committed to in a slide deck, the cost of changing course has grown substantially and your manager's ego is now partly invested in the outcome. If you see a direction forming that you think is flawed, speak up in the conversation where it's forming. A question or a gentle "have we thought through..." in a planning meeting is worth ten objections after the decision is public.

Lead with curiosity, not conclusion. "I think this is wrong" is a hard thing to receive, especially if you're right. "I want to make sure I understand the reasoning here, because from where I sit there's a constraint that might affect the plan" is the same substantive concern, opened in a way that invites dialogue rather than triggering defensiveness. You're not softening the message — you're creating the conditions where it can actually land.

Be specific about what you're observing. Vague discomfort isn't pushback, it's noise. "I'm not sure this will work" gives your manager nothing to evaluate. "The timeline assumes the data migration can be completed in two weeks, but based on our last migration it took five, and I want to flag that risk before we commit" gives them something concrete to weigh. Specific concerns get taken seriously. Vague concerns get noted and ignored.

And then let it go. If you've made your case clearly and your manager has heard it and still wants to proceed, that's the decision. You can note your disagreement — briefly, once, clearly — and then execute with full commitment. "I still have concerns about the timeline but I understand your reasoning and I'll make sure we're tracking the risk closely" is a perfectly adult sentence. What you don't get to do is undermine the direction, quietly sandbag the effort, or perform a told-you-so retrospective. You made your case. They made the call. That's how this works.

When Your Boss Was Your Peer

There is a specific and awkward flavor of managing up that deserves its own section: when your manager was, until recently, sitting at a desk near yours, arguing with you about the best way to handle a merge conflict.

This happens constantly. Organizations promote from within. Peer relationships become reporting relationships. And then both parties have to figure out how to behave, which turns out to be genuinely hard.

From the managing-up side — when you've been promoted and your old boss is now your boss again, or when your boss is someone you knew as a peer at a different level — the trap is overcorrection. You either become hyperformally deferential, which is awkward and reads as fake, or you maintain the old peer dynamic so thoroughly that you can't give them honest upward information because it feels like tattling or self-promotion in a context that used to be collegial.

Neither works. What does work is naming the dynamic once, briefly, and then building a new operating agreement on top of it. Something like: "This is a different relationship than we had before, and I want to make sure I'm giving you what you actually need from me now, not what made sense when we were both ICs." That conversation, which takes about four minutes, prevents months of misaligned expectations.

If the awkward dynamic runs the other direction — your boss was recently your peer and is now your manager, perhaps after a reorg — the challenge is that they may feel the need to prove their authority, or conversely may overcorrect toward informality because they don't want to seem like the role changed them. Your job is to make it easy for them to manage you well. Give them the information they need without forcing them to ask. Be explicit about what you need. Don't test them by seeing if they'll hold you accountable the way they hold others accountable. They will be tempted to give you slack because you used to be colleagues. Don't take the slack. You'll both regret it.

What Your Boss Actually Needs to Hear

The last thing on this topic, and possibly the most counterintuitive: your boss needs to know when you don't know something.

There is enormous pressure, especially early in a management role, to project competence at all times. You are supposed to have answers. You are supposed to be in control. You are the manager. And so the instinct, when something is uncertain or unclear or genuinely outside your experience, is to figure it out quietly and only reveal it once you've resolved it.

This has a name in the research literature. A 2011 study by Alison Wood Brooks and Francesca Gino at Harvard found that people systematically underestimate how positively advice-seeking is perceived — they assume asking for help signals weakness, when in fact it signals competence and self-awareness. Your manager, in most cases, would rather hear "I haven't dealt with this before and I want to think through it with you" than watch you quietly mishandle something that required experience you hadn't yet built.

Zhuo frames this as one of the core skills of a good manager: knowing the edges of your own knowledge and being comfortable naming them. "Great managers," she writes, "know when they're in over their head and aren't too proud to say so." That applies upward as much as it applies to the team below you.

There is something clarifying about being a new manager that is easy to lose and worth holding onto: nobody expects you to have everything figured out. Your manager almost certainly doesn't expect perfection. What they need from you is transparency — about what's working, what isn't, where you need guidance, and where you're genuinely confident and don't need them in the weeds. Give them that, consistently and clearly, and you will be one of the easiest people on their team to manage.

That is, as it turns out, a genuinely high bar.

The Agreement You Haven't Had Yet

Before you close this chapter and move on, one concrete ask: schedule a working-agreement conversation with your manager. Not a performance check-in, not a status update — an explicit conversation about how the two of you operate together.

Ask them: How do you like to receive information — written or verbal, synchronously or async? What's your preference on frequency? Are there topics you want to hear about proactively, even before I have answers? What would make you feel like I'm doing my job well, and what would make you feel like I'm dropping the ball?

And share the same from your side. What kind of feedback is useful to you. How you prefer to work through disagreements. What you need from them in terms of context and organizational visibility to do your job.

This conversation will feel slightly awkward and take approximately twenty minutes. It will save you months of misread signals, missed expectations, and accumulated friction. Almost no one has it. The managers who do have it tend to look, to outside observers, like they have an unusually good relationship with their boss. They don't have a better relationship. They have a cleaner interface.

That's the frame. Your boss is a user. Design for them deliberately, give them what they need to function, and surface the problems before they become the problems that define you.

After that hallway conversation with Clara — the one after the Derek meeting, the one I deserved — I went back to my desk and wrote a one-page document. It had three sections: what's green on my team right now, what's yellow and being managed, and what I wasn't sure about yet. I sent it to her that afternoon. It was not polished. Some of it was incomplete. She replied within ten minutes: "This is exactly what I need. Can we do this every Friday?"

We did it every Friday for the next two years. Not because I was performing diligence, but because I'd finally understood what the job actually required. You're not just managing your team. You're managing the relationship that makes managing your team possible.

One more thing Clara said, in that same hallway, that I've thought about often since:

"The goal isn't for you to look capable. The goal is for us to actually be capable. Those are not the same thing."

She was right. They're almost never the same thing.

CHAPTER XI

Politics, Briefly (Because You Can't Pretend It Doesn't Exist)

The word "politics" has a way of shutting down useful conversation before it starts. Say it in a team meeting and you'll get one of three reactions: knowing grimaces from the veterans, righteous eyerolls from the people who've decided they're above it, and a studied blankness from whoever in the room is currently playing the game and doesn't want you to notice.

That third person is the one worth watching.

I'm going to keep this chapter shorter than the others. Not because organizational politics is simple — it isn't — but because I've spent over a decade watching managers succeed and fail, and I've noticed something consistent: the people most consumed by workplace politics, the ones who analyze it constantly and see it lurking behind every setback, are frequently the ones generating the most of it. They've turned "politics" into an explanatory framework that accounts for every loss and locates all the dysfunction somewhere outside themselves. It's a very comfortable theory. It's often almost completely wrong.

So this is a minimal viable map. How decisions really get made. How to build relationships before you desperately need them. And where the line is between being organizationally literate and becoming someone your team quietly doesn't trust anymore.

Use it honestly.

What Politics Actually Is (And Why It's Not a Character Flaw)

Strip the drama off the word and what you're left with is a structural condition, not a moral one. Politics is what happens when people with different priorities compete for finite resources inside an organization.

That's all it is.

Budget is finite. Engineering headcount is finite. Time on the CEO's calendar is finite. Space on the product roadmap is genuinely, stubbornly finite — no amount of quarterly planning ceremonies has ever changed this. When multiple teams need more of any of those things than exist, someone has to decide who gets them. The process by which that decision gets made, and all the maneuvering that surrounds it, is organizational politics. It didn't start because a VP decided to be difficult. It emerged the same way traffic congestion emerges: from many people making individually rational decisions, producing collectively messy outcomes that nobody designed and nobody fully controls.

Andy Grove made a distinction in High Output Management that's worth putting here at the front: the difference between "position power" — authority that flows from your title — and "knowledge power" — authority that flows from expertise, information, and relationships. His argument was that in modern organizations, especially in knowledge work, title-based authority is considerably weaker than it looks on the org chart. The VP who nominally controls the budget may actually be heavily influenced by the senior engineer who built the system everyone depends on, the director who's attended every planning cycle for seven years and knows where the bodies are buried, and the two-year employee who has somehow become the informal hub of institutional memory that nobody officially designated her for.

Org charts describe reporting lines. They do not describe where influence actually lives.

Understanding that gap — the difference between the chart and the actual influence network — is organizational literacy. It's a job requirement for managers, in the same category as reading a P&L or running a hiring process. The malignant version of politics, the kind that drives talented people to update their LinkedIn on a Tuesday afternoon, is what happens when someone starts gaming that influence network for personal gain instead of organizational outcomes. When a peer buries your project in a steering committee not because it's a bad project but because it competes with theirs for the same executive's favor. When someone withholds information from a cross-functional partner to maintain a strategic edge. When someone takes credit for collaborative work in a skip-level meeting and calls it "visibility."

That's not politics as a structural condition. That's bad behavior with a polished alibi.

The distinction matters practically. If you call all organizational influence "politics" and opt out entirely, you've refused to do a significant part of your job. If you call the malignant version "just how it works," you've handed yourself a permission slip to corrode the people around you. Name them differently. Respond to them differently.

The Meeting Before the Meeting

Here is something nobody puts in the job description: the meeting where the decision appears to happen is usually not the meeting where the decision actually happened.

The formal structures — steering committees, quarterly planning sessions, weekly staff meetings — are largely where decisions get ratified, explained, and documented. The actual decision was made two or three conversations earlier. In a hallway. In a Slack thread between two people who trust each other's judgment. In a pre-read that one party sent and the other actually read. In a 1:1 that happened to land on Tuesday afternoon when someone's calendar had an opening.

This isn't corruption. It's how humans function. People arrive at contentious meetings having already thought about the issue, talked to people they respect, and formed a tentative view. The meeting tests that view against other views, and the view that arrives better-prepared, with more stakeholder alignment behind it, typically wins. What looks like a surprising decision in a big room is almost always the product of one party doing its pre-work — building informal alignment before the formal forum — and another party assuming the

meeting would be the first conversation.

Camille Fournier identifies this as one of the most consistent gaps between technically excellent individual contributors and effective managers in The Manager's Path. The IC is accustomed to their work speaking for itself. The manager has to learn — and it takes longer than it should — that work, even excellent work, needs an audience that's been prepared to receive it. You can write the clearest technical proposal in company history and lose the room in the first ten minutes because two of the five decision-makers heard a different framing from someone else that morning at nine.

The practical implication: before any consequential meeting — a roadmap negotiation, a headcount ask, a reorg proposal, a budget presentation — map the room. Who's in it? What do they care about most right now? Who has already formed a strong view, and what shaped that view? Whose buy-in is load-bearing, meaning their skepticism will pull others into skepticism, their support will draw others into support?

Then have conversations before the conversation.

Not to manipulate the outcome. To understand what you're walking into and to surface real objections while you still have time to address them. There is a meaningful difference between "I talked to Marcus before the meeting because I wanted to understand his concerns about the data pipeline risk" and "I talked to Marcus before the meeting so he'd anchor the room in my favor when Priya pushed back." The first is preparation. The second is coalition-building for its own sake. Both happen constantly. Only one serves the organization.

Julie Zhuo observes in The Making of a Manager that the most effective managers she studied were consistently the ones who made it easy for people to understand what they were trying to accomplish and why — not through persuasion as a technique, but through clarity as a default. When your intentions are legible and people can see what you're actually optimizing for, pre-meeting alignment feels like collaboration. When intentions are murky, the exact same behavior reads as maneuvering.

Worth sitting with. The version of pre-work that feels slimy is usually the version where the goal is unclear, or where the person doing it suspects their proposal wouldn't survive an honest open evaluation. If your idea is good and you're genuinely trying to serve the organization's interest, walking it through stakeholders before a formal vote feels natural. If it wouldn't survive that honesty, you don't have a political problem. You have a proposal problem.

Build Relationships Before You Need Them as Favors

The single most reliable piece of political advice I can offer you is also the least manipulative one: build relationships with people outside your team before you have a crisis that requires their help.

This sounds obvious. First-time managers almost universally ignore it for the first twelve to eighteen months.

Understandably so. The early months of managing are genuinely absorbing. You're learning your people, learning the work at a level of detail you didn't need before, learning where all the decisions actually get made. The cross-functional relationships — with the product manager your team ships alongside, the data

team that supports your metrics, the legal team that reviews your contracts, the finance partner who owns your budget model — those get treated as purely transactional. You reach out when you have a problem. You're pleasant in the large forums. You probably don't know their managers' names until something goes wrong and you need an urgent favor.

Here's what happens when you build a relationship exactly at the moment of need: you're asking for goodwill from someone who has no accumulated reason to extend it to you, under conditions that are already stressful enough that neither of you is operating at your best. The conversation is loaded with urgency and stakes and the implicit subtext of one person needing something from another. That's survivable, but it's harder than it has to be. Every time.

Contrast that with the manager who has lunch with her product counterpart twice a quarter — no standing agenda, just "what are you working on, what's genuinely frustrating you, what are you worried about six months from now." Who sends a message to the data science team lead after a blog post she found interesting, not because she needs an analysis run but because she's actually curious about their work. Who knows the name of the senior attorney who handles the contracts her team generates, because she introduced herself at a company all-hands fourteen months ago and has emailed three times since with nothing more urgent than a question and a thank-you.

That manager's asks land differently. They feel like requests from a colleague rather than cold transactions from someone who just remembered she needed something. The goodwill is real, and it moves faster and with less friction when you actually need it.

Jeffrey Pfeffer, who has studied organizational power at Stanford Business School for four decades, has a characteristically unsentimental formulation: the time to build a network is when you don't need one. His research found consistently that managers who invested in low-stakes relationship-building had better outcomes during high-stakes moments — not because they were more skilled at manipulation, but because they had access to better information, more honest feedback, and more generosity from peers who'd already formed a positive view of them.

The practical implementation doesn't require becoming someone who performs friendliness as a competitive strategy. Pick three to five people outside your team whose work intersects meaningfully with yours. Put a thirty-minute recurring conversation on the calendar, quarterly. No agenda required. Ask what they're working on. Listen without an ulterior motive. Share what you're working on. That's the whole practice. It is not dramatic. It compounds quietly over eighteen months and then one day you need to move fast on something that crosses functional lines and you discover you already have exactly the relationship you needed because you built it during a nothing period on an ordinary Tuesday.

Information Is the Actual Currency

Grove's framework about position power versus knowledge power points toward something that took me embarrassingly long to absorb: in most organizations, information is the real currency of influence.

Not in a surveillance sense. In a practical one. The person who knows what the company's actual strategic priorities are — not the ones on the all-hands slides but the ones the CEO genuinely lies awake thinking about — has leverage that nobody handed her in writing. The person who understands which technical debt will cause the most significant failures in the next twelve months, and can explain the stakes to the executives who control the refactor budget, has influence that exceeds her title by a notable margin. The person who knows three weeks in advance that the enterprise sales team is about to close a deal that will shift the product roadmap — because she's maintained a relationship with that sales leader — can plan her team's work while everyone else is about to scramble.

Grove's prescription was relentless information gathering. He described his own practice of management by walking around — not as a touchy-feely management philosophy but as an intelligence-collection methodology. The 1:1s, the casual drop-bys, the questions you don't technically need to ask: all of it aimed at building a picture of the organization that formal reporting structures and weekly status emails cannot provide.

This matters for your effectiveness in a specific, concrete way. A lot of what looks like "political maneuvering" from the outside is actually information asymmetry. One person knew which way the decision was going to land because they'd had the right conversations. They raised the right objection at the right moment because they understood the full context. They won the headcount allocation because they framed their ask in terms of the company's current actual priorities, not last year's planning document priorities, because they'd done enough organizational circulation to know the difference.

You can't gather that information from inside your team's silo. You have to be in the organizational bloodstream — in meetings, in conversations, in the informal forums that most managers skip because they're busy — not because you're running an intelligence operation, but because it's how any manager builds an accurate model of the environment their team is operating inside. Fournier puts it plainly: you cannot make good decisions for your team without understanding what the organization around your team is actually trying to accomplish. That's not political wisdom. That's basic navigation.

Where the Line Is

Everything in this chapter so far describes legitimate organizational engagement. Building relationships. Understanding informal influence networks. Preparing for consequential decisions. Gathering information. All of that is fine. Most of it is required.

Here is where it goes wrong.

The slide from effective navigation to actual maneuvering — the kind that quietly corrodes your integrity until one day you're someone you wouldn't have recognized at the start of the job — is almost never a single dramatic choice. It's a sequence of small ones, each individually justifiable, that accumulate into something you'd struggle to explain to someone who trusted you.

It usually starts with a moment where the honest move is costly. Your team's project gets deprioritized in planning and you suspect it's partly because the other team's manager has a longer relationship with the product VP. The honest response is to understand the reasoning, make your case through legitimate channels, and adjust your plan if the decision holds. The tempting response is to start seeding doubt about the competing project in conversations it can't defend itself in. Those are different moves. The second one works occasionally, in the short run, and now you've started something.

Or: you have information that would genuinely help a peer in another function, but sharing it would also strengthen their position relative to yours in an upcoming resource discussion. Sharing it is correct and also the move that builds the kind of trust that compounds over years. Sitting on it gives you a possible tactical edge for one quarter and leaves you with a colleague who will, eventually, notice the pattern.

The test I've found most useful isn't a formal ethical framework. It's simpler. Would you be comfortable if your team could hear exactly how you described this situation to your own manager? Not the polished version. The actual version, with the actual reasoning, said out loud.

Bad political behavior fails that test quickly. "I didn't mention the data quality issue to the product team because I didn't want them to delay the launch before we could demonstrate our value" — say that sentence out loud and listen to it. "I positioned our team's completeness as stronger than it actually was in the steering committee because I knew Akira's team was about to make a competing pitch" — same. Those sentences reveal their own problem the moment they're spoken in plain English.

Zhuo makes an observation in The Making of a Manager that applies here with uncomfortable precision: the behaviors that most consistently destroy team trust are rarely the dramatic ones. They're the pattern of small choices — slightly self-serving framings, information withheld selectively, credit distributed unevenly — that accumulate until people can sense you're optimizing for something other than the shared goal, even when they can't point to a single clear act of misconduct. They feel it before they can name it. And once they feel it, it takes years to undo.

Your reputation as a manager is built across hundreds of small choices, not one or two big ones. That's either encouraging or alarming depending on where you currently stand.

The People Who Create It

I promised to keep this chapter short, and here's why I meant it.

The managers most consumed by organizational politics — the ones who have elaborate running theories of who's aligned with whom, who see political maneuvering as the explanation for every setback, who could draw you a detailed influence map at a moment's notice — are frequently the ones generating the most friction in the system. Not always. Sometimes the analysis is accurate, the game genuinely is rigged in a specific direction, and the person calling it out is doing everyone around them a service.

But often, the manager with the most sophisticated theory of the organizational power structure has been trying to play that game for two years and losing. The theory is an explanatory framework that locates all the dysfunction outside themselves. Their ideas don't win because of politics. The wrong people get promoted because of politics. Resources flow in the wrong direction because of politics. It can be internally very coherent. It can miss the point almost completely.

The antidote isn't naivety. It's staying anchored to outcomes rather than games. What does your team need to do excellent work? What does the organization need from your team right now? What relationships, resources, and information do you need to make that possible? Start from those questions and you will find yourself doing the appropriate amount of organizational navigation — enough to be effective, not so much that it becomes your primary activity and your primary identity.

A rough calibration: if you're spending more than ten or fifteen percent of your working week thinking about the political dimensions of your situation — who's aligned with whom, how to position something for maximum influence, what the relationship subtext of some decision means — you've probably crossed from organizational awareness into a loop that isn't serving you or your team. The work is still the work. The team is still the actual unit of output. Politics is context. Not content.

Grove was direct about the purpose of a manager's job: the combined output of all the people they're responsible for. Not the quality of their executive relationships considered in isolation. Not the elegance of their organizational positioning. The output of the people. Everything else, including every piece of advice in this chapter, exists in service of that.

So. Stay there. Do good work. Be someone other teams genuinely want to work with because you're honest about tradeoffs, useful under pressure, and straightforward about what your team can and cannot do. Build relationships because you're curious about what other people are building, not because you're managing a ledger of obligations owed. Tell the truth in rooms where telling the truth is uncomfortable — which will happen, repeatedly, throughout your career, and is worth doing every single time.

You will not win every resource conversation. You will not always be in the room where the consequential decision happens. You will sometimes lose to someone who outmaneuvered you, and the only honest question afterward is whether you lost because you were outplayed or because they had the better argument.

Either way. Do the work.

The politics resolves itself, over enough time, in the direction of the people who actually know what they're doing and can be trusted when something real is at stake.

CHAPTER XII

When You're Stuck: Coaches, Mentors, and Books That Actually Help

There's a specific kind of Tuesday afternoon — you'll recognize it when it arrives — where you close your laptop, stare at the middle distance for about thirty seconds, and think: I have absolutely no idea what I'm doing.

Not the generic imposter-syndrome hum that lives in the background of most management roles. Something sharper. A personnel situation that has no clean answer. A team dynamic that has quietly curdled and you can't find the seam to pull. A conversation you've now rehearsed in your head eleven times and it goes badly in all eleven versions. You're not incompetent — you got this job for real reasons — but nothing in your previous experience gave you the manual for this particular problem, and you're out of plays.

The move most first-time managers make at this point is to suffer quietly. To carry the thing alone, run it through their head in the shower, maybe vent obliquely to their partner at dinner, and then show up the next morning hoping the situation has somehow resolved itself. It almost never has.

This chapter is about building the support infrastructure before you need it. Mentors. Coaches. Peer networks. Books that are actually worth your time. None of this is soft or optional — it's the operational layer that keeps experienced managers functional and that almost nobody tells first-timers about until they've already burned a year reinventing wheels that other people solved a long time ago.

The Infrastructure Problem Nobody Mentions

Here's what the org chart doesn't show you: every senior manager you admire has a network of people they call when things go sideways. A mentor they've known for a decade. A peer they grab coffee with who runs a team at a different company. Maybe an executive coach. A small, trusted group of people who have seen enough management situations that they can say "oh, that's a classic," and then tell you what actually happened when they tried option A versus option B.

You don't have that yet. And you won't build it by accident.

The other thing nobody mentions: your boss, however good they are, has a limited capacity to help you with certain kinds of problems. Anything involving your boss's own behavior is immediately off the table. Anything involving a peer of yours requires discretion that complicates the conversation. And anything that might make you look less competent — even if they'd tell you otherwise — you're probably not bringing to them at full candor. That's not a failure of trust. That's just the structural reality of a reporting relationship.

So you need other sources. Let's go through them.

Mentors: What They Are and What They're Not

The word "mentor" gets used to describe about six different relationships, most of which aren't mentorship. Your skip-level who checks in once a quarter isn't your mentor. The senior leader who gave a talk at a conference and said "feel free to reach out" definitely isn't your mentor. And the person your company assigned you in a formal mentorship program might or might not be your mentor, depending entirely on whether you actually have chemistry with them and they have time to invest.

Real mentorship is a relationship where someone with more experience in roughly your domain gives you ongoing, honest feedback on real situations — not a performance of advice-giving, but actual engagement with the specific problem in front of you. It requires frequency (probably once a month at minimum to stay warm), honesty on both sides, and enough contextual knowledge that they're not just offering generic frameworks you could have found in a book.

The distinction that matters most, and that people confuse constantly, is mentor versus sponsor. A mentor gives you advice. A sponsor gives you opportunities. These are different people doing different things for different reasons.

Your mentor says: "Here's how I'd think about that performance conversation." Your sponsor says: "I'm putting your name in for that cross-functional leadership role." Mentors help you get better. Sponsors pull you up. You need both, and confusing them leads to the frustrating situation of having excellent mentorship relationships that never move your career and wondering why.

For the purposes of this chapter, we're mostly talking about mentors — people who can help you think through management problems. How do you find them?

The honest answer is: it's awkward and most people don't do it well. The most common mistake is treating mentorship as a favor you're asking a stranger for, which it kind of is, and which is why the ask goes badly. Cold LinkedIn messages to senior leaders you've never met have roughly a 2 percent success rate and generate mostly polite noncommittal responses.

What actually works is developing relationships with people before you need them. Former managers you respected. Senior peers from previous roles. People you've worked with on cross-functional projects who you genuinely liked. The relationship has to already exist in some form. Then the mentorship conversation isn't "will you be my mentor," which is a weird and pressure-laden ask, but something more like: "I'm working through something at work and I'd really value your perspective — could we grab thirty minutes?" That's a low-stakes first ask. If it goes well, you ask again. Over time it becomes a thing.

One constraint worth naming: try to find mentors who aren't at your current company if you can. Someone inside the company is useful in some ways but hamstrung in others — they have their own relationships with the people and politics you're describing, which makes it hard for them to be fully candid. An external mentor

who has no stake in your specific org can say things your internal mentors won't.

Executive Coaching: When It's Worth the Cost

Executive coaching costs money. Real money — somewhere between $300 and $600 an hour on the reasonable end, higher if you're engaging someone with a serious pedigree. That price tends to end the conversation for most first-time managers before it starts, which is partly why so many of them are flying blind.

But before you dismiss it entirely: check whether your company pays for it. Many companies above a certain size have coaching budgets that go underutilized because people don't know to ask. Your HR business partner is the right person to ask. The answer might be yes, with a process attached.

If you are paying out of pocket, the calculus is this: coaching is worth the cost when the situation you're navigating has a dollar value attached that exceeds the cost of help. A team that's underperforming costs you time, stress, and reputation. A promotion you don't get costs you career trajectory. A key person who quits because of a management dynamic you couldn't see and fix costs the company recruiting fees and months of productivity. Viewed that way, ten sessions with a good coach looks cheap.

The question is what makes a coach good. The coaching industry has essentially no quality control — anyone can hang out a shingle — so you're almost entirely reliant on referrals and your own judgment. A few things to screen for: Do they have domain knowledge, meaning they've managed people in roughly your kind of environment, or managed people who manage people? Do they ask you useful questions or do they mostly validate you? Do they have a model or framework they work from, and does it make sense to you? And critically: are they willing to push back, or do they just reflect your thinking back at you in a slightly more organized form?

That last one is the real test. A coach who only validates is expensive journaling. The good ones make you uncomfortable in ways that turn out to be useful.

Coaching is especially valuable in three situations. First, when you're in a genuine leadership crisis — your team is falling apart, your boss has lost confidence in you, and you need an external perspective badly and quickly. Second, when you're making a significant transition — stepping up to managing managers for the first time, or taking over a team in a new function — and you want someone to help you think through the shape of the new role before you're in it. Third, and this one surprises people, when things are basically going fine but you've plateaued and can feel it. A coach is not just for crisis management. The best use of coaching is often preemptive.

Peer Manager Networks: The Thing That Actually Saved Me

In year two of managing, I was deep in a team morale problem that I couldn't see clearly because I was too close to it. My 1:1s were fine. The work was getting done. But something was off — the energy in team

meetings had gone flat, a couple of people were job-hunting (I found out later), and I kept attributing it to external factors. Quarterly planning stress. The economy. The project being in a boring phase.

I brought it to my manager, who gave me reasonable advice that didn't quite fit. I called a mentor, who asked good questions but didn't have enough context about my specific team to be concrete. I read a management article that told me to "foster psychological safety," which, yes, but how.

What actually helped was a conversation with a peer manager I barely knew — someone who ran a team at a different company, no overlap with mine, no stake in the politics — who said, almost immediately: "That sounds like what happened to my team when I stopped running retrospectives. When did you last close the loop on something the team flagged?"

I hadn't. In eight months, we'd raised about six team-level process issues and I had addressed exactly none of them, not because I was ignoring them but because I'd gotten busy and forgot to circle back. That was the problem. Forty-five minutes, fixed in a week.

The peer found me through a small Slack group that had been started by someone I knew from a previous job. Twelve managers, all at different companies, roughly the same level. No agenda, no structure, just a #management-problems channel where people posted things they were wrestling with and got candid responses from people who had no reason to be anything other than honest.

That group was the most practically useful resource I had in year two. Not by a small margin. It was genuinely irreplaceable in ways that books, mentors, and managers couldn't be, because it was specific, real-time, and horizontal — nobody was managing anybody else, so the power dynamics that distort so many conversations didn't exist.

Building one isn't complicated. It requires maybe eight to fifteen people at similar career stages, mix of companies and industries (pure single-industry groups become echo chambers), someone willing to do the light admin of inviting people and nudging the channel when it goes quiet, and a clear shared understanding that the conversations stay in the group. That last piece matters. People won't be candid about real personnel situations if they're worried the details will travel.

If you already have a version of this — a few peers you text when things get weird — you basically have the informal version. Making it slightly more intentional, adding a few more people with different backgrounds, and giving it a home (Slack, Discord, a group text, doesn't matter) makes it dramatically more useful. The diversity of the group is what makes it work. You want someone who manages engineers, someone who manages a creative team, someone who manages a sales org — because the problems look different on the surface but are the same underneath, and the cross-pollination of frameworks is what gets you unstuck.

The Reading List (Annotated and Honest)

There are approximately four thousand books about management and leadership. Most of them are not worth your time. A depressing number are airport-bookstore business books that dress up one medium insight as

three hundred pages of chapters. Here are the three that actually matter, and specifically what each one is useful for.

High Output Management — Andy Grove

This book was written in 1983 and is still the most technically precise account of what management actually is and what it's for. Grove was the CEO of Intel and ran his management program like an engineer: there are inputs, there are outputs, there are leverage points. His core claim — that a manager's output is the output of the team they manage and the teams they influence — sounds simple until you internalize it and realize it changes every decision you make about how to spend your time.

The chapter on the managerial meeting is worth the price of the book by itself. Grove distinguishes between process-oriented and mission-oriented meetings with a clarity that most management books never achieve, and his framework for deciding which decisions should be made at which level of the hierarchy is still the clearest version of that problem I've encountered.

What it's not: warm. Grove doesn't have much patience for the emotional texture of management — the difficult conversations, the psychological complexity of feedback, the way people's histories affect their behavior at work. He's writing about systems, not souls. So you need to read it alongside books that address what he leaves out.

Read it when: you're feeling lost about what your actual job is. When you're doing too much of the wrong things. When you need someone to explain management as a discipline with rigor and without sentimentality.

The Making of a Manager — Julie Zhuo

Zhuo was a twenty-five-year-old designer at Facebook when she became a manager for the first time, which means she made every first-time mistake under a level of scrutiny most of us will never face and wrote about it with unusual candor. The book has a warmth and specificity that Grove doesn't, and it's particularly strong on the early-stage questions: how do you know if you're doing it right, how do you build trust with people who were your peers, what does a good 1:1 actually look like in practice.

Her observation that a manager's defensiveness is the single biggest killer of team psychological safety is the kind of sentence that stays with you. She's good on feedback, good on hiring, good on the inner experience of management — the self-doubt, the way your team's bad days become your bad days, the odd loneliness of the role.

If Grove is the architecture, Zhuo is the interior design. She fills in the human spaces he leaves empty.

What it's not: particularly useful for managing managers, or for the organizational complexity that comes later. This is a book about being a first-level manager of individual contributors, which is exactly what you need right now and less useful as you move up.

Read it when: you're in your first year and need someone to normalize how hard it is. When you're struggling with feedback conversations. When you need to remember that other smart people also found this genuinely

difficult.

The Manager's Path — Camille Fournier

Fournier's book is structurally different from the other two: it's a career map, organized by management level, from tech lead through VP of Engineering. Which means you can read the whole thing as a roadmap for where you're going, or read just the chapters that apply to where you are right now.

For first-time managers, the chapters on mentoring and managing individuals are where to start. Fournier is direct in a way that's bracing — she doesn't soft-pedal hard truths about how managers routinely avoid difficult conversations, fail to set clear expectations, and then act confused when performance goes sideways. Her framework for the difference between a manager's job (creating opportunity and clarity) and the employee's job (seizing it) is one of the cleanest articulations of that boundary I've read anywhere.

She's also the best of the three on the structural side of people management — performance frameworks, career ladders, what good engineering culture looks like. Coming from a technical background, she's less useful if you're managing a creative or commercial team, but the management-of-people principles translate reasonably well even if the technical specifics don't apply.

What it's not: as emotionally immediate as Zhuo. Fournier is precise and pragmatic and covers enormous ground, which means she sometimes moves through nuanced situations faster than you'd like.

Read it when: you want to understand the full shape of a management career and see where first-level management fits. When you're dealing with a career development conversation with a direct report and need a framework for thinking about paths. When you're about to start managing someone more senior than you and need to think carefully about how that works.

A short note on what's missing from this list

I've deliberately left off several famous management books. If you're waiting for Good to Great or The Five Dysfunctions of a Team or Radical Candor — they're not here. Radical Candor has good ideas but is more usefully read as a blog post than a book; the core framework is genuinely helpful and the padding around it is not. The Kim Scott TED talk is probably sufficient. Five Dysfunctions is a business fable, which is a genre I find structurally irritating; its insights are real but I'd rather get them from Lencioni's summary than the extended novel format. And Good to Great was a serious research project that later replication attempts have been unkind to — some of the "great" companies in the original study failed fairly dramatically in the decade after publication.

Read what's on the list first. There's enough there.

What Asking for Help Actually Requires

None of this support infrastructure works if you don't actually use it. Which sounds obvious, but the reason most managers don't build it is the same reason they don't ask for help when they're stuck: asking for help

feels like admitting you don't know what you're doing, and when you're new to authority, that feels like a dangerous exposure.

Here's the reframe: the managers who don't ask for help are the ones who don't know what they're doing and broadcast it through their team's dysfunction. The managers who build support networks and use them are the ones who get better faster, handle hard situations with more skill, and hold onto their good people longer. Asking for help is not a sign of weakness. It's how the job actually works for the people who are good at it.

There's research behind this, not just intuition. A 2019 study published in the Journal of Applied Psychology found that managers with higher quality peer networks — specifically characterized by trust, informality, and diversity of perspective — made significantly better people management decisions and had teams with lower turnover. The mechanism isn't complicated: other people see your blind spots. That's it. Nobody has enough perspective on themselves to see all of their blind spots alone.

Grove understood this in a structural way — he built Intel's management culture around information-sharing and what he called "managerial leverage," the idea that the right input at the right moment multiplies the output of everything downstream. A mentor who saves you three months of trial and error on a team dynamics problem is a leverage point. A peer who names your pattern before it becomes a crisis is a leverage point. A book that gives you a framework you were searching for is a leverage point.

You are not an island managing in isolation. The mythology of the self-sufficient leader who figures everything out through force of will is almost entirely fiction, and the few real examples are not people you want to emulate — they tend to leave bodies behind them.

Assembling Your Stack

Here's the practical version. By the end of your first year, you should have:

A mentor you talk to at least once a month. Someone external if possible, with enough domain overlap to be concrete. You don't need more than one to start, and one good one beats three mediocre ones.

At least five peer managers you have an honest relationship with — people at your level, not your direct reports or your boss, who you can text "is this as weird as I think it is" and get a straight answer. Five sounds like a small number. For most first-time managers it requires active effort to build even that.

A reading stack of three books. The three above. In that order if you're starting from nothing: Zhuo first (orientation), Grove second (structure), Fournier third (map). Two hundred and fifty hours of management problems went into those three books. They are not a substitute for experience but they are a significant head start.

And a standing question you ask yourself, maybe once a month: what am I currently stuck on, and have I told anyone? If the answer is "yes, I have a serious problem and I haven't mentioned it to a single person," that's a flag. Not because you're weak. Because you're doing the one thing that reliably keeps the problem exactly the size it is.

The wall everyone hits is real. The Tuesday afternoon stare into the middle distance — I've had it dozens of times, and I'll have it again. The difference between managers who develop and managers who calcify is not that the first group hits fewer walls. It's that they built the infrastructure to get unstuck.

Build it before you need it. You're going to need it.

CHAPTER XIII

Burnout: Yours and Theirs

There's a version of burnout that announces itself. You know the story: some high-powered executive works ninety-hour weeks, misses their kid's entire childhood, and one Tuesday morning can't get out of bed. The body simply stops. The drama is legible. Friends gather. A Netflix documentary is not out of the question.

That version exists. It's just not the version most managers experience.

What actually happens is quieter, and for that reason, considerably more dangerous. You don't stop going to work. You don't stop answering Slack messages. You show up, you run the meetings, you write the performance reviews. You do everything on the list. But somewhere in the space of a few months, you stop doing it well, and then — almost without noticing — you stop caring whether you're doing it well. The 1:1s get shorter. The questions get blunter. The team meeting that used to have some real back-and-forth starts feeling like a status report you're presiding over with the low-level dissociation of someone watching their own Zoom call from a slight distance.

That's the thing nobody warns you about. Burnout in managers doesn't look like a breakdown. It looks like Tuesday.

Why This Role Is Structurally Exhausting

First, let's be honest about the conditions you're working in, because managers who burn out often blame themselves before they examine the setup.

Andy Grove wrote in High Output Management that a manager's output is the output of their organization. Simple enough. But what that means in practice is that your job never fully resolves. An engineer ships a feature and closes the ticket. A designer delivers the mockup. There's a done state. There's a moment when you can look at the thing and say: finished. Management doesn't have that. Your team's performance, the health of your relationships, the slow accumulation of trust — these don't close. They're open tabs, always. Every Friday you leave without having finished anything in the way a craftsperson finishes something.

Add to that what Camille Fournier describes in The Manager's Path: the manager's role is emotionally absorptive in a way that individual contributor roles usually aren't. Your team's anxiety becomes your anxiety. Their interpersonal conflicts land on your desk. The senior engineer who is quietly looking at LinkedIn, the junior developer who cried in a 1:1 last week, the product manager who keeps cc'ing your skip-level on emails — all of that lives in your head, not theirs. You are a container for other people's stress, and containers have limits.

Christina Maslach, the psychologist whose work on burnout has defined the field for four decades, breaks it into three components: emotional exhaustion, depersonalization (a clinical term for the cynicism that kicks in when you stop seeing the people in front of you as fully human), and reduced sense of personal accomplishment. Notice that none of those are physical. You can be sleeping fine, exercising, eating reasonably well, and still score high on all three. The body is fine. It's the orientation toward the work that has curdled.

Management amplifies all three risk factors in ways that most job descriptions conveniently omit.

Emotional exhaustion is built into the role — you are in the business of other people's feelings and performance, by definition. Depersonalization creeps in when you've given the same feedback conversation for the fifteenth time, or when you look at your calendar on Monday morning and see wall-to-wall meetings and feel nothing except the low-grade desire to fake a dental emergency. Reduced accomplishment arrives when your manager's manager asks what you've been working on and you genuinely struggle to articulate it — because what you've been working on is invisible: conversations, context, the slow maintenance of an environment where other people can do their best work.

None of this makes you weak. It makes you normal, in a role that doesn't always come with the support to stay healthy inside it.

Year Three

I want to tell you about six months in my third year as a manager, because I think it's more useful than any framework I could give you in the abstract.

By that point I had a team of eight engineers, a product roadmap I mostly believed in, and a manager who was stretched thin across three teams and trusted me to run independently. On paper, things were fine. Objectively, things were fine. The team was shipping. Nobody had quit in over a year.

But I had started to notice something in my 1:1s. I wasn't curious anymore. I was going through the motions — running through the same five questions, nodding at the same answers, walking away with notes I didn't really read back. One of my engineers, Marcus, had mentioned twice that he wanted to talk about moving toward a tech lead role. Both times I said something supportive and vague and changed the subject because I didn't have the bandwidth to think through what that path actually looked like on our team. He brought it up a third time. I did the same thing. He started job hunting two months later, which I found out in his resignation 1:1, and I remember sitting there thinking: I knew this was coming and I did nothing, and the reason I did nothing is that I was too depleted to do the thing that is literally my job.

That was the moment I named it. Not because of some crisis, not because anything had exploded. Because I had failed someone quietly, in slow motion, and I had watched myself do it.

Julie Zhuo writes in The Making of a Manager that one of the clearest signals of a manager in trouble is when feedback stops being specific. When you hear yourself giving people generic encouragement — "you're doing

great," "really solid quarter" — it's often because you've lost the thread. You're not paying close enough attention to give the real thing, and you've made peace with the substitute. That sentence hit me like a diagnosis. I had been writing performance check-ins for three people that said, essentially, nothing. Warm, HR-appropriate nothing.

The six months that followed Marcus leaving were the ones where I actually looked at what was happening and made some changes. I'll come back to what those were. First, though — the signals.

The Early Warning Signals in You

The trouble with spotting your own burnout is that it impairs the very faculties you'd use to spot it. You become less reflective at precisely the moment reflection matters most.

So here are the behavioral tells, not the feelings. Because feelings are unreliable reporters. Behaviors aren't.

Your preparation drops first. You start going into 1:1s without reviewing your notes from the last one. You show up to the team meeting without a real agenda, figuring you'll improvise. You pull together that presentation for the leadership review at 11 PM the night before instead of two days before. This isn't laziness — it's the first sign that the mental energy required to prepare feels disproportionate to the reward. The reward used to feel real. Now it feels abstract.

You get impatient in ways you hide. Someone asks a question you've answered before, and you feel a flash of something — not quite irritation, but its close cousin. You answer anyway, pleasantly. But the flash was there. It's getting faster. In meetings you find yourself checking Slack under the table, which you would have judged yourself for six months ago.

The optimism goes flat. Managers who are running well tend to be, at minimum, oriented toward possibility. Not delusionally positive, but genuinely interested in solving the next problem. When you notice that you're reacting to new challenges with a low-grade "here we go again" instead of anything resembling curiosity, that's worth paying attention to. Gallup's 2023 State of the Global Workplace report found that managers show lower wellbeing scores than individual contributors in most industries — and that one of the key markers is a decline in what they call "thriving" orientation, the sense that things can get better.

You stop developing people. This one is subtle. It looks, from the outside, like things are running fine. The team is executing. But if you look at the last ninety days and can't name a concrete thing you did to grow each person's skills or expand their opportunities, that's a signal. Fournier puts it plainly: the manager's core job is creating conditions for growth, and that job is the first one to go when a manager is depleted. It requires discretionary energy. Depleted people don't have discretionary energy.

You avoid the hard conversations. Not dramatically. You don't refuse to have them. You just let another week go by. And then another. The feedback you owe someone has been sitting in your mental queue for a month and you keep finding reasons to delay. This is important, because avoidance has a compounding cost — the longer you wait, the more awkward it becomes, the more you have to address the gap between when you

noticed and when you said something. The avoidance itself becomes a thing you're managing.

If three or more of these are true right now, this chapter is for you specifically.

The Early Warning Signals in Your Team

Here's the part managers almost never see coming: your burnout is not private. It transmits.

Your team reads you constantly and accurately, often more accurately than you read yourself. They notice the shorter answers, the distracted look, the 1:1 that ended twelve minutes early because you "had to jump on something." They don't necessarily know the word for what they're observing, but they know something is off, and they will adapt to it — not always in ways that are good for anyone.

Watch for these signals in your team:

Decreasing candor in meetings. If you notice that your team's discussions have gotten shallower — less debate, faster consensus, fewer people pushing back — it may not be that everyone has suddenly aligned. It may be that they've stopped trusting the container. A disengaged manager creates a team that self-censors, because candor requires believing the response will be thoughtful. When people sense you're checked out, they give you the version of the conversation that requires the least from you.

An uptick in working-around behavior. When your team starts solving problems by going directly to other teams, other managers, or your own manager instead of routing through you — not because you've explicitly encouraged autonomy, but because it's faster and lower friction — that's a sign. You've become a bottleneck people are routing around rather than a resource they're drawing on.

One-on-one attendance gets flaky. People start rescheduling your 1:1s at a higher rate. Sometimes this is legitimately scheduling chaos. Often it's a rational response to meetings that have stopped being worth the thirty minutes. They're not doing it to punish you. They're doing the math.

Your highest performers get quiet. This one is the real alarm bell. Your best people have options. They have external visibility, they have recruiters in their LinkedIn DMs. They're also, typically, the most sensitive to management quality, because they know what good looks like. When your A-players start going heads-down in a way that looks like disengagement rather than focus — when the energy they used to bring to team discussions goes absent — start asking questions fast. Not about whether they're okay (too vague), but specific ones: what's been frustrating lately? What's getting in their way? What would make the next quarter feel different from this one?

You might not love the answers. Get them anyway.

The Recovery Framework (No Sabbatical Required)

I want to be direct here: most recovery from manager burnout doesn't require quitting, taking a month off, or any dramatic restructuring. It requires four things done in sequence. Small things, honestly. But the sequence

matters.

Step one: Name it without negotiating with it.

The first response most managers have when they recognize burnout is to immediately argue with the diagnosis. It's not that bad. Lots of people have it harder. I just need a good weekend. This is the managerial version of the same avoidance instinct that delays feedback conversations. Stop. Name the thing clearly, at least to yourself: I am burning out. I am not doing my job well right now. The people I'm responsible for are not getting what they need from me.

That's not self-flagellation. It's accurate data collection. You cannot fix what you haven't named.

Step two: Do a brutal 15-minute calendar audit.

Get a blank piece of paper. Draw three columns: Energy drain, Energy neutral, Energy return. Then go through your recurring commitments — every meeting, every standing obligation — and put each one in a column.

The purpose is not to eliminate the drains. Most of them are legitimate and non-negotiable. The purpose is to see the actual distribution, because managers in burnout almost always have an imbalance that they've normalized. When I did this exercise in year three, I found that I had eleven recurring meetings per week. Two of them gave me any real energy. Seven were neutral. Two were actively depleting in ways I'd been absorbing without naming. One of those two — a cross-functional sync that had outlived its original purpose by about six months — I was able to cancel outright after a single conversation with the other meeting owner. We both wanted out. Neither of us had said so.

One meeting. It sounds trivial. It wasn't.

Step three: Find one person to be honest with.

Not your team. Not your direct reports — they're not the right container for this. But someone: your manager, if the relationship allows it, or a peer manager you trust, or the peer slack group I described in the previous chapter.

Julie Zhuo is clear on this in The Making of a Manager: the loneliness of management is real and it's a structural feature, not a personal failing. You are expected to hold a lot for your team without putting your own chaos back into the system. That's appropriate. But it means you need somewhere for the pressure to go, because it goes somewhere whether you direct it or not. If you don't have a peer you can be honest with, that's its own problem to solve — but the immediate step is to find one conversation where you can say "I'm struggling" without performing competence.

The reason this matters is simple: isolation accelerates burnout, and visibility slows it. Just having someone else know you're in it changes the internal experience. You go from carrying it alone to carrying it with a witness, and that is a meaningful difference.

Step four: Recommit to one thing at full quality.

Not everything. One thing. Pick the person on your team whose development you've been most neglecting and spend genuinely good time on their next 1:1. Prepare for it the way you used to prepare when you thought preparation mattered. Come in with real questions, not the comfortable ones. Listen to the answers like you're going to be tested on them later.

The reason this step works is that burnout is partly an erosion of the feedback loop between effort and meaning. When you phone it in, you get a phoned-in result, and that confirms the subterranean belief that nothing you do really matters anyway. One good 1:1 doesn't cure burnout. But one moment where you do the real thing and see it land — where you ask a real question and get a real answer and say something that genuinely helps — that starts to rebuild the case that the work is worth doing. You are, in a modest way, re-engaging the reward circuitry of the role.

Then do it again the next week.

Their Burnout Is Different From Yours

Your team members can also burn out. You knew that. But the way it presents in individual contributors is different from how it presents in managers, and the response is different too.

For ICs, burnout usually has a cleaner precipitating cause: sustained overwork, a project without clear progress markers, repeated unrecognized effort, or a deep mismatch between the work they're doing and the work they want to be doing. Fournier's observation in The Manager's Path is that the manager's job in these cases is first to diagnose honestly — because "you seem tired" is not a diagnosis — and then to adjust either the workload or the meaning, or both.

Adjusting workload is the obvious move. It's also frequently insufficient on its own, because for most skilled people, burnout isn't primarily about hours. It's about return on investment. They're spending something — their attention, their creativity, their care — and they're not seeing it come back in the form of progress, recognition, growth, or impact. That's the real drain.

Ask them directly: "What part of your work right now is giving you energy? What part is taking it?" Then actually do something with the answer, even if what you can do is limited. Being seen is not nothing. Being told "yes, that thing is genuinely tedious and I can't change it but I want you to know I see it" is not nothing. It's not everything, but it matters.

The failure mode here is happy-talking your team through burnout. "Things are going to get better after the launch" is only useful if it's true and if "after the launch" is soon. If the launch is in five months and the honest forecast is that Q4 will be equally demanding, say so. Informed people can pace themselves. People who've been told to expect relief that doesn't come stop trusting their manager's read on reality.

That trust, once lost, is slow to rebuild.

A Note on Prevention

Everything so far has been about recognizing and recovering. What about not getting there?

Three practices, briefly, because they're more useful as habits than as emergency interventions:

Do a monthly energy audit, not an annual one. The same three-column exercise takes fifteen minutes. Most managers do something like this after a crisis. The ones who sustain best do it preventively. Catching the drift before it becomes a rut is significantly easier than reversing the rut once you're in it.

Protect one thing that makes you good at this. Every manager I've known who stayed sharp over multiple years had at least one non-negotiable practice that anchored them — a weekly run, a standing reading hour, a journal, a monthly dinner with a mentor. Not because wellness habits are magic, but because they create a regular check-in with yourself that doesn't go through the performance of being fine. You are alone with yourself for sixty minutes, and you have to report honestly on how things are going. That's harder to sustain when you're burning out, which is exactly when it matters most.

Get granular with your manager. Grove's model in High Output Management treats the manager-manager relationship as a critical information channel — the skip-level, the regular sync, the calibration conversation. That channel should include your actual state, not just your team's state. If your manager only ever hears that things are fine, they'll believe you, and you'll be on your own with the things that aren't. This isn't weakness. It's data hygiene.

I lost Marcus. He went to a competitor, had a genuinely good career there from what I can tell, and probably doesn't think about me much. But I think about him in the context of that period, because his departure was the clearest evidence of a cost I had been running up without putting it on the ledger.

The six months after that I got better, not because anything dramatic happened, but because I started naming things. I did the calendar audit. I told a peer manager I was struggling. I had one genuinely good 1:1 with someone I'd been phoning in on, and then another, and somewhere in there the work started feeling like work again instead of like weather I was standing in.

That's not a triumphant ending. It's a functional one. Which, in year three of managing people, might be the best kind.

You're going to hit a wall. Maybe you already have. The question is whether you name it early enough to recover before the people who depend on you pay the price that Marcus paid.

They're watching your 1:1 quality. They're counting on your curiosity. Even when yours runs thin — especially then — they deserve the version of you that shows up prepared.

That's the hardest part of this role. It is also, if you can stay honest with yourself about it, the one you'll be most proud of getting right.

CHAPTER XIV

The Kind of Manager You Want to Be Remembered As

Nobody is going to build a statue of you because you shipped a Q3 roadmap on time.

That's not cynicism. It's the first honest thing worth saying in a closing chapter, because most first-time managers spend the early years optimizing for the wrong scoreboard — hitting delivery milestones, avoiding HR escalations, keeping their skip-level meetings smooth. Those things matter. They're not nothing. But they are not what former direct reports remember when they run into each other at industry conferences and someone asks, "Who was the best manager you ever had?"

Think about that question for a second. Who was the best manager you ever had? Not the most organized. Not the one who ran the tightest process. The best one. Chances are, you remember a person who saw something in you before you saw it in yourself. Someone who gave you a project that was probably too big for you at the time and then didn't take it back when you struggled. Someone who told you a hard truth in a way that made you better instead of making you smaller.

That's the bar. And it's worth deciding, right now, whether you're actually trying to clear it.

What You're Actually Optimizing For

Andy Grove wrote in High Output Management that a manager's output is the output of the teams under their influence — not just the people they directly manage, but everyone they touch. He meant it practically: your job is to multiply, not to perform. But there's a longer version of that idea that Grove didn't quite spell out, which is that the most durable output a manager produces isn't the quarterly results. It's the people they shaped.

Camille Fournier puts a finer point on it in The Manager's Path. The best technical managers, she argues, leave behind a trail of engineers who became better engineers — and sometimes better managers — because of the environment that person created. It compounds. One great manager produces a handful of people who go on to create dozens more. The organizational math on that is staggering if you actually sit with it.

Julie Zhuo frames it slightly differently in The Making of a Manager: "The best thing you can do for the people you work with is to help them do their best work." Not do the work for them. Not protect them from difficulty. Help them find whatever version of their best work they're capable of, and then get out of the way.

Three books, three frameworks, one underlying idea: your job is to make other people bigger.

The problem is that almost nothing about the day-to-day of management reinforces this. Your calendar is full of things that feel urgent. You're reacting. You're in meetings designed to produce other meetings. And the slow, patient work of developing a person — asking the right question in a 1:1, noticing when someone is ready to be stretched, advocating for them in a calibration conversation they never see — that work produces no immediate feedback signal. You don't know it's working until years later, often from an email you weren't expecting.

The Credo Exercise Nobody Assigned You

Here's something most management books don't ask you to do: write down what you actually believe.

Not your company's values. Not your team's OKRs. Your beliefs. About people. About work. About what you owe the people who report to you.

Call it a management credo if you want a name for it. Leadership philosophy if that sounds less dramatic. Call it nothing at all, just write it. The act of writing it forces you to be specific in ways that thinking about it never does.

Here's what a credo is not: it's not a list of platitudes. "I believe in people," and "I value transparency" are the management equivalent of a LinkedIn headline — technically true, meaning nothing. A real credo has stakes. It says something you could fail to live up to. It names things you actually believe, not things that sound good in an all-hands.

A few prompts to get you started:

What do you believe about people's capacity to grow? This is not a rhetorical question. Some managers operate from a quiet assumption that people are roughly what they are — that a mid performer will always be a mid performer, that someone's ceiling is visible from the first six months. Other managers genuinely believe people are mostly in the wrong context and that the right environment can unlock something different. These are not equally valid positions. The research is pretty clear: Gallup's long-running workplace studies consistently find that managers who hold high expectations of their direct reports actually get higher performance, independent of the employee's prior track record. Your belief here shapes everything about how you run 1:1s, how you assign projects, and what stories you tell about your team in calibration rooms.

What are you willing to fight for? Not in the abstract — specifically. Are you willing to fight for someone's promotion when HR says the budget's frozen? Are you willing to push back in a meeting when a peer manager's plan is going to overload your team? Are you willing to tell your own boss that a direction is wrong, with your name on that disagreement? These are the moments that define your credibility with your team more than any team offsite or Slack message ever will. Your people notice when you advocate for them and when you don't. They remember.

What do you want a direct report to say about you five years from now? Not what you hope they think. What would they actually say, if someone asked them at a conference, completely unprompted, when they had no

reason to be kind? Write the specific sentences. Make them concrete. "She pushed me to take on the platform migration when I was convinced I wasn't ready" is a real sentence. "He was a great leader" is filler.

Write the credo before you need it. Because the moments that test who you are as a manager don't announce themselves. They show up in the last ten minutes of a 1:1 on a Thursday, when someone says something that lands harder than expected. They show up in the calibration meeting when you could take the easy road and say nothing. They show up when you're running on empty in month eight and someone needs you to show up fully.

The credo is what you reach for when the situation is ambiguous and you don't have time to think.

What Great Managers Actually Do Differently

Google ran a major internal study on team effectiveness starting in 2012, eventually published under the name Project Aristotle. They wanted to know what made some teams consistently outperform others. They expected the answer to be about composition — smarter people, more complementary skills. What they found instead was that the single most predictive variable was psychological safety: whether team members felt safe to take risks without fear of humiliation or punishment. And psychological safety, more than any other variable, is created or destroyed by the manager.

Not by the team. By the manager.

The specific behaviors that create it are not exotic. They include: admitting when you don't know something, following up on feedback you receive instead of letting it disappear, being visibly wrong sometimes without getting defensive, and treating people's ideas as worth engaging seriously even when you disagree. These are behaviors, not traits. You can learn them. You can practice them. You can fail at them and try again next week.

Zhuo makes an observation in The Making of a Manager that should probably be printed on a card and left on every first-time manager's desk: the thing that kills psychological safety faster than anything else is manager defensiveness. Not incompetence. Not bad strategy. Defensiveness. The manager who can't hear a critical observation without explaining why it's wrong. The manager who takes feedback as an attack. That manager, over time, stops getting honest information from their team — and then makes worse and worse decisions on an increasingly distorted picture of reality.

The good news is the inverse is also true. A manager who genuinely receives feedback — who says "that's useful, thank you, here's what I'm going to do with it" and then actually does something — builds an unusual kind of trust. It signals that the channel is open. People start sending more honest signals because they've seen that honesty doesn't get punished.

You don't have to be perfect. You have to be genuine.

The Line Between Standards and Control

There's a version of "making people better" that is actually a cage.

The manager who is deeply invested in everyone's growth but whose definition of growth is "doing things the way I would do them." The manager who gives people big projects but micromanages the execution. The manager who talks about trust but reviews every pull request and edits every doc before it goes to a stakeholder.

Fournier addresses this directly in The Manager's Path when she talks about the transition from senior individual contributor to manager. The hardest thing, she says, isn't learning the new management skills. It's letting go of the direct control over the output. You have opinions about how the code should be architected. You have opinions about the right way to run the product review. You're probably right. But if you impose those opinions constantly, you're not developing your people — you're just borrowing their hands while they borrow your brain.

The managers people remember as great were almost universally the ones who created room. Room to try things. Room to be wrong in low-stakes situations so that you built the skill to be right in high-stakes ones. Room to own something fully — not just do the work, but own the decision and the consequence.

That's the distinction between standards and control. Standards say: here is what good looks like, and here is why. Control says: here is exactly what you will do. The first one scales. The second one doesn't, and it tends to slowly hollow out everyone around you until you're surrounded by people who are waiting to be told what to do.

If that's not what you want — and it shouldn't be — then the practice is to notice when you're reaching for the pen, and ask yourself whose development is being served by grabbing it.

Ryan's Desk Drawer

I want to tell you about a note I've kept for four years.

His name was Marcus. I managed him for eighteen months early in my career, when I was twenty-nine and he was twenty-six and both of us were figuring out what we were doing. He was a backend engineer, genuinely talented, and I was his first real manager after a brief stint under a team lead who'd treated 1:1s like status reports and code reviews like performance evaluations.

I made a lot of mistakes with Marcus. I gave him feedback too late on a project that stalled, when I should have raised the flag six weeks earlier. I pushed back on a technical approach he'd championed for three weeks and was right to champion, because I hadn't understood it fully and confused my own uncertainty with his being wrong. I spent the first four months of his time on my team watching what he did rather than asking him what he wanted.

The thing I got right, eventually, was this: I stopped managing Marcus's work and started managing Marcus. Meaning, I asked about his three-year ambitions, not his current ticket velocity. I put him in a room with the VP of Engineering for a project review even though he'd never presented to that level before, and I prepped

him for an hour beforehand instead of doing the presentation myself. When he shipped something exceptional, I said so out loud, in Slack, with specifics, because recognition that happens in private doesn't compound.

When he told me he was leaving — he had a role offer to lead a small infrastructure team at another company, his first management role — I remember sitting in the conference room after he left and doing the math on whether I should have done more to retain him. I decided I shouldn't. What he was going to was right for him. I helped him write the thank-you note to our CEO and walked him out on his last day.

About fourteen months later, I got an email. Marcus, now at his third company since leaving mine, was a VP of Infrastructure at a Series B startup I'd never heard of. The email was three paragraphs. The last one said: "I've been thinking about what made the difference for me early on and I think it was mostly that you treated me like someone who was going to go do big things before I believed it about myself. I don't know if that was intentional but it mattered more than I can explain."

That email is printed out and in my desk drawer. Paper. Which is absurd given that I work in tech. But it's there.

I've thought about why I keep it, and I think it's this: it's the only artifact I have that tells me whether I did the job correctly. Not whether the team shipped on time. Not whether my calibration ratings were defensible. Whether I actually did what I was there to do, which was to help someone become more than they were when they arrived.

That's the job.

Writing the Last Line First

There's a planning technique sometimes called "beginning with the end in mind" — it gets attributed, usually, to Stephen Covey, though the underlying logic is older than he is. You decide where you want to arrive before you map the route.

Apply it here.

Imagine a scenario, as concrete as you can make it. Five years from now, someone who reported to you — maybe Marcus, maybe someone you'll hire next quarter — is at a conference. An industry friend asks them, casually, "Who taught you the most about working?" They could say a lot of things. They could mention a book, or a course, or a peer they collaborated with. But say they mention you. What do you want the specific words to be?

Not "she was a great manager." Not "he had really good processes." The specific words.

Write them down. Seriously — close this book for five minutes and write them. Because those sentences, the ones you just wrote, are your actual management credo, stripped of all the framing. They're what you're really aiming for.

Now ask yourself whether the way you ran this week's 1:1s moves you toward that, or away from it.

This is not a technique for feeling good about yourself. It's a technique for detecting drift early, before it hardens into a pattern. The manager who wanted to be remembered for taking chances on people but hasn't given a junior engineer a stretch assignment in six months is drifting. The manager who wanted to be remembered for honesty but has been softening every piece of critical feedback since a tense conversation in February is drifting. The check-in question — "does this week's behavior match the thing I said I believed?" — is free and available every Sunday night.

The Long Game

Grove said management is the highest-leverage activity available to an organization. He meant it in the operational sense: one manager, well positioned, affects the output of a dozen or twenty or a hundred people. The return on a manager getting better is enormous.

But there's a longer version of the leverage argument, and it's the one that gets you out of bed on the bad weeks.

Every person you develop well goes somewhere. They go to other teams, other companies, other industries. They carry forward some specific thing that happened in your 1:1s, or in a feedback conversation you gave them, or in a project you trusted them with before they were technically ready. They bring that thing to the people they manage, if they become managers. They bring it to the teams they join. The specific way someone was seen, heard, and challenged by a good manager — that doesn't disappear when the reporting relationship ends. It replicates.

This is not abstract. There are managers working today who were shaped in practical ways by managers who are dead. Not through formal training. Through the habits and approaches that got passed forward — the specific way someone was told their work mattered, the question that reframed how they thought about a problem, the moment of trust that arrived slightly before they felt ready for it. That's what compounds over decades.

You are one link in a chain that started before you and will continue long after you're not managing anyone anymore. That's either motivating or vertiginous, depending on your temperament. It should probably be both.

The Last Thing Worth Saying

This book has given you a lot of frameworks. Feedback models, 1:1 structures, hiring rubrics, career conversation cadences. Those frameworks matter. They're not decoration. A consistent 1:1 practice genuinely changes team outcomes. Structured interviews genuinely reduce bias in hiring. Written performance reviews that link to future growth genuinely open conversations that vague ratings close.

But frameworks are scaffolding. They hold the structure up while you're building. The building itself is something else.

The building is the set of specific choices you made when the framework didn't tell you what to do. The time you went to bat for someone in a meeting where it cost you something. The time you said "I don't know" in front of the whole team instead of performing certainty. The time you gave a person the project you wanted to run yourself because their development needed it more than your ego did.

Nobody writes those choices down in a performance review. No rubric captures them. But the people on the receiving end remember them with startling clarity, sometimes for the rest of their careers.

Marcus's email has three paragraphs. The one that gets me, the one I re-read occasionally when I'm having a bad management week, is not the thank-you. It's the line that says: "before I believed it about myself."

That's the window. You see something in someone before they can see it. You tell them. You back it up with actual opportunity, not just encouragement. And then you wait, and they grow into it, and five years later you get an email from a VP at a company you've never heard of.

That is, if you do it right, the whole job.

Decide now what kind of manager you want to be. Write the credo. Read it back. Ask whether today's choices match it.

Then go do the work.