Manager's Playbook

Jonny Dubowsky

Manager's Playbook

Original post: https://raw.githubusercontent.com/ksindi/managers-playbook/master/README.md

Heuristics for effective management: principles, one-on-ones, coaching, feedback, making decisions, meetings, hiring, onboarding, announcing change.

Principles

1. Always be aware of what's going on in your team and product.
2. Know the architecture just as well as anyone else — and stay current with the tech stack.
3. Schedule 1-1s with your direct reports on a weekly basis. Schedule skip-levels on a monthly basis.
4. Be customer-focused. Understand how your products are used in the wild. For example, join sales and support calls.
5. Set aggressive but achievable goals. Work backwards by focusing on the outcomes you want to achieve.
6. When giving advice or feedback, encourage ownership by asking open questions.
7. Have a bias for action and decision-making over planning and consensus.
8. Be the team coach and cheerleader. Celebrate success often and reinforce positive behavior.
9. Know how to differentiate between [reversible and irreversible decisions][making-decisions].
10. Ensure every report is aware of the top priorities of the team, organization and company.
11. Be the example. Only preach what you practice.
12. No task is beneath a manager. Get your hands dirty even if it's not coding:

One on ones

1. Never skip one on ones. It's the best platform for receiving and giving feedback. Most teammates value it and usually when they don't it's because they haven't seen one conducted well.

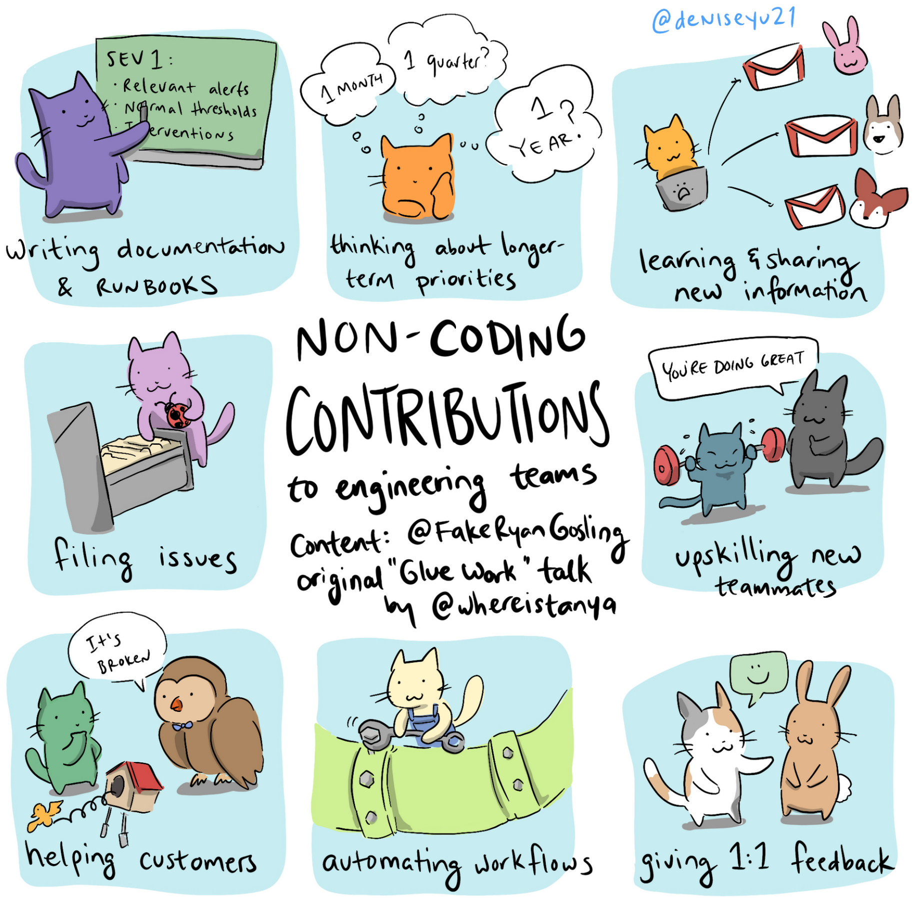

@deniseyu21
SEV 1:
· Relevant alerts
· Normal thresholds
· Interventions
writing documentation & RUNBOOKS
1 MONTH
1 quarter?
1 YEAR?
thinking about longer-term priorities
learning & sharing new information
NON-CODING CONTRIBUTIONS to engineering teams
Content: @FakeRyanGosling original "Glue Work" talk by @whereistanya
filing issues
YOU'RE DOING GREAT
upskilling new teammates
It's BROKEN
helping customers
automating workflows
giving 1:1 feedback

Figure 1: Non-Coding Contributions

2. Aim for weekly one on ones.
3. Focus on 5 topics:
    - Predictability: create routine, set expectations, normalize change.
    - Ownership: offer options, clarify ownership, give more responsibilities.
    - Purpose: clarify the big picture value and importance of their tasks.
    - Progress: create milestones, share wins, celebrate progress.
    - Belonging: team culture and management.
4. Questions to help frame each topic. “On a scale of 1-10 how would you rate:…”
    - Predictability: How clear do you feel about what’s expected of you?
    - Ownership: Your satisfaction with decision power and direction?
    - Purpose: How much your work makes a difference for the team?
    - Progress: The sense of progression each week?
    - Belonging: Your feeling of connection to the team?
5. Additional questions to ask on a less frequent basis:
    1. Motivators:
        1. Which part of your work is most fun?
        2. What’s not fun about working here?
        3. What are the biggest time wasters for you each week?
    2. Long term goals:
        1. What skills do you want to improve?
        2. What career path are you looking for?
        3. Who in the company would you be excited to learn more from?
        4. What parts of the business would you like to be more involved in?
    3. Organization awareness:
        1. What don’t you like about the product?
        2. What’s the biggest opportunity that we’re missing out on?
        3. What do you see as your top 3 priorities this quarter? The team’s? The org’s?
        4. If you were CEO, what would you do differently?
    4. Manager’s role:
        1. What could I do to support you better?
        2. If you were me, what are 1 to 3 things you would change?
        3. How do you feel about the amount of feedback you are getting?
        4. I need feedback. What are two things that I can do differently?
        5. What’s one thing we could do to improve our way of collaborating?
    5. Priorities:
        1. “What are your top priorities this week?”
        2. “What will success look like?”
        3. “What are obstacles?”
6. Encourage your direct reports to bring up topics in 1-1.
7. See Getting more from your one on ones for more thoughts on effective one on ones.

8. There isn't one best management style. Figure out how someone wants to be managed in your initial one on one:
    - What did you like about your previous manager? What didn't you like?
    - What do you like to see from a manager?
9. One on ones where the report is also a manager are typically more "business" where the focus is on strategy, team health and project alignment.

Coaching

1. Default to open questions: ask questions starting with **what**, **how**, **who** instead of closed-ended ones like **do**, **have**, **is** to invite conversation and give ownership over a problem.
    - "What questions do you have?" is better than "Do you have any questions?".
    - "Why do you think this is the right approach?" is better than "Is this a good idea?".
    - Respond with "What are your thoughts so far?" when asked "What should I do?"
2. Summarize what the person is saying so you're both on the same page and are pinpointing the problem.
    - "It sounds like there are two issues, x and y. Which should we focus on first?"
3. Figure out how to make this meeting productive.
    - "What's the next step?"
    - "How should we track this?"

Feedback

1. Be prompt, ideally providing feedback the same day of the event that prompted it.
2. Get buy-in about providing feedback and reduce mystery:
    - "Do you have 10 minutes to talk about this morning's stand up?"
    - "Can I share some thoughts with you about how we've been working together?"
3. Don't "pad" negative feedback by beginning with compliments - it gives mixed signals.
4. Be specific even if it's positive feedback.
    - "Good job!" (no)
    - "I like the initiative you took to reduce the service's memory footprint. It shows ownership and leadership."
5. Focus on data and not behavior:
    - "I noticed you didn't address any of the comments made in your last three PRs"

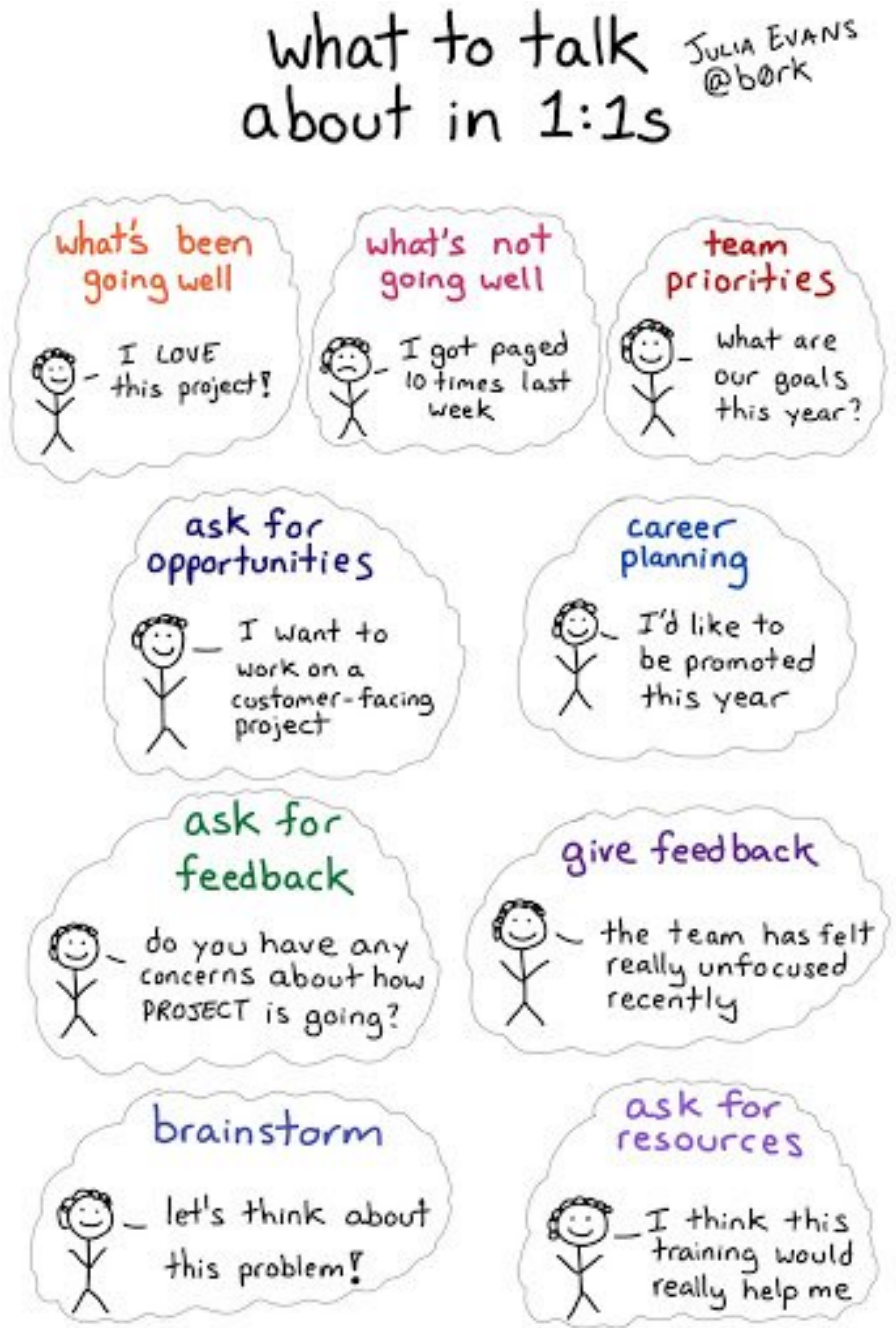

what to talk about in 1:1s
JULIA EVANS @b0rk
what's been going well
I LOVE this project!
what's not going well
I got paged 10 times last week
team priorities
what are our goals this year?
ask for opportunities
I want to work on a customer-facing project
career planning
I'd like to be promoted this year
ask for feedback
do you have any concerns about how PROJECT is going?
give feedback
the team has felt really unfocused recently
brainstorm
let's think about this problem!
ask for resources
I think this training would really help me

Figure 2: What to talk about in one-on-ones

    - “I noticed that you didn’t pick up the ticket I asked you to do”
    - “I noticed your last feature release didn’t have any tests”
    - “Your code is buggy” (no)
    - “You are always late” (no)
6. Talk about why this matters and who it’s affecting:
    - “I mention it because it’s important we work as a team”
    - “I mention it because the ticket I assigned you is critical to this quarter’s roadmap”
7. Figure out together how to fix the problem:
    - “What do you think of our process?”
    - “How do you see it?”
8. Agree on an action plan:
    - “How do we ensure we don’t miss a ticket due date next time?”
    - “What are a couple of actions you could take right now?”
    - “What are our action items?”
9. Highlight positive patterns (remember to be specific).
    - “I like when you take initiative in cleaning up code because it shows initiative and ownership.”
    - “It’s great to see you teach X about Y so that they’re. That’s a positive trait of a senior engineer.”
10. Replay instinctive reactions to help frame the conversation

Thinking strategically

1. What would you do with one more person?
2. How is your team moving the needle? Are you focusing on the right things? How do you know the features you’re building will benefit the customer?
3. What are your product’s mission and tenets?
4. What are the company’s top priorities this year? Where should the company be three years from now?
5. What are your “rocks” and “pebble” projects this quarter?
6. What pillars is your team driving and in what way?
7. What are your team’s pain points? How can you move 2x faster?
8. What “dogs not barking” do you worry about? What are areas in your team you worry about?

Making decisions

1. Determine if the decision is reversible vs. irreversible.
    - Reversible decisions can easily be changed. Examples: changing stand up time, contributing guidelines.
    - Irreversible decisions cannot be changed without significant rework. These decisions should take longer and be documented and discussed. Examples: architecture changes, language decision, data models.

- A rubric on technology decisions by Sam Newman:

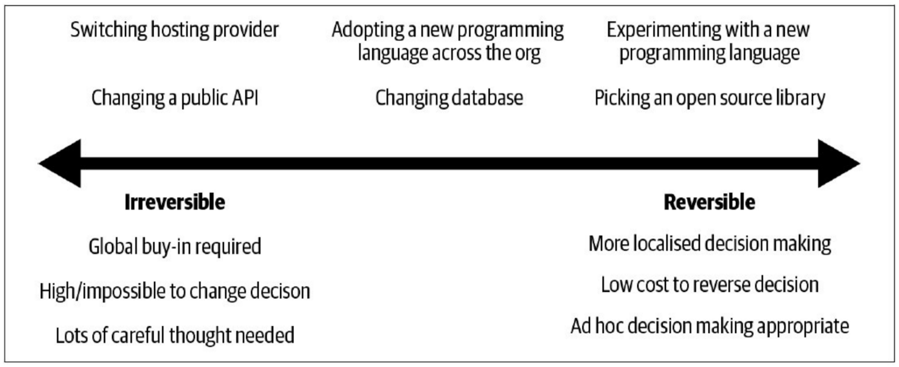

Switching hosting provider
Adopting a new programming language across the org
Experimenting with a new programming language
Changing a public API
Changing database
Picking an open source library
Irreversible
Reversible
Global buy-in required
High/impossible to change decison
Lots of careful thought needed
More localised decision making
Low cost to reverse decision
Ad hoc decision making appropriate

Figure 2-3. The differences between Irreversible and Reversible decisions, with examples along the spectrum

Figure 3: Irreversible and reversible decisions

1. Whenever there is disagreement, focus on the intended outcome of the decision and make sure the team is aware of your reasoning.
   - "While database X is better, I want us to standardize on one stack so that it's easier to maintain."
2. If someone disagrees with a reversible decision, set a date to revisit that decision with the team. Ideally you also have metrics to define the success of that decision.
   - "I understand your concerns. Let's revisit this in a month and see where we stand."
   - "We're tracking X now, let's revisit next quarter if it improves with these changes."
3. If someone disagrees with an irreversible decision, give them the opportunity to present their case. Regardless, everyone should be aware the decision is ultimately yours and the team needs to disagree and commit wholly to the decision made.
4. Document your decisions so that you can refer to why they were made and the tradeoffs your team faced.

Coding

Avoid coding in the critical path. Tips from Charity Majors: 1. Authoring a feature? (no) 2. Covering on-call when someone needs a break? (yes) 3. Diving on the biggest project after a postmortem? (no) 4. Code reviews? (yes) 5. Picking up a p2 bug that's annoying but never seems to become top priority? (yes) 6. Insisting that all commits be gated on their approval? (no) 7. Cleaning up the monitoring checks and writing a library to generate coverage? (yes)

Ticket and PR process

1. Set contributing guidelines for the team.
2. PRs should always be prioritized. Aim for review SLA of 1 hour.
3. Automate opinions like style with linting or code formatters like black.

Email

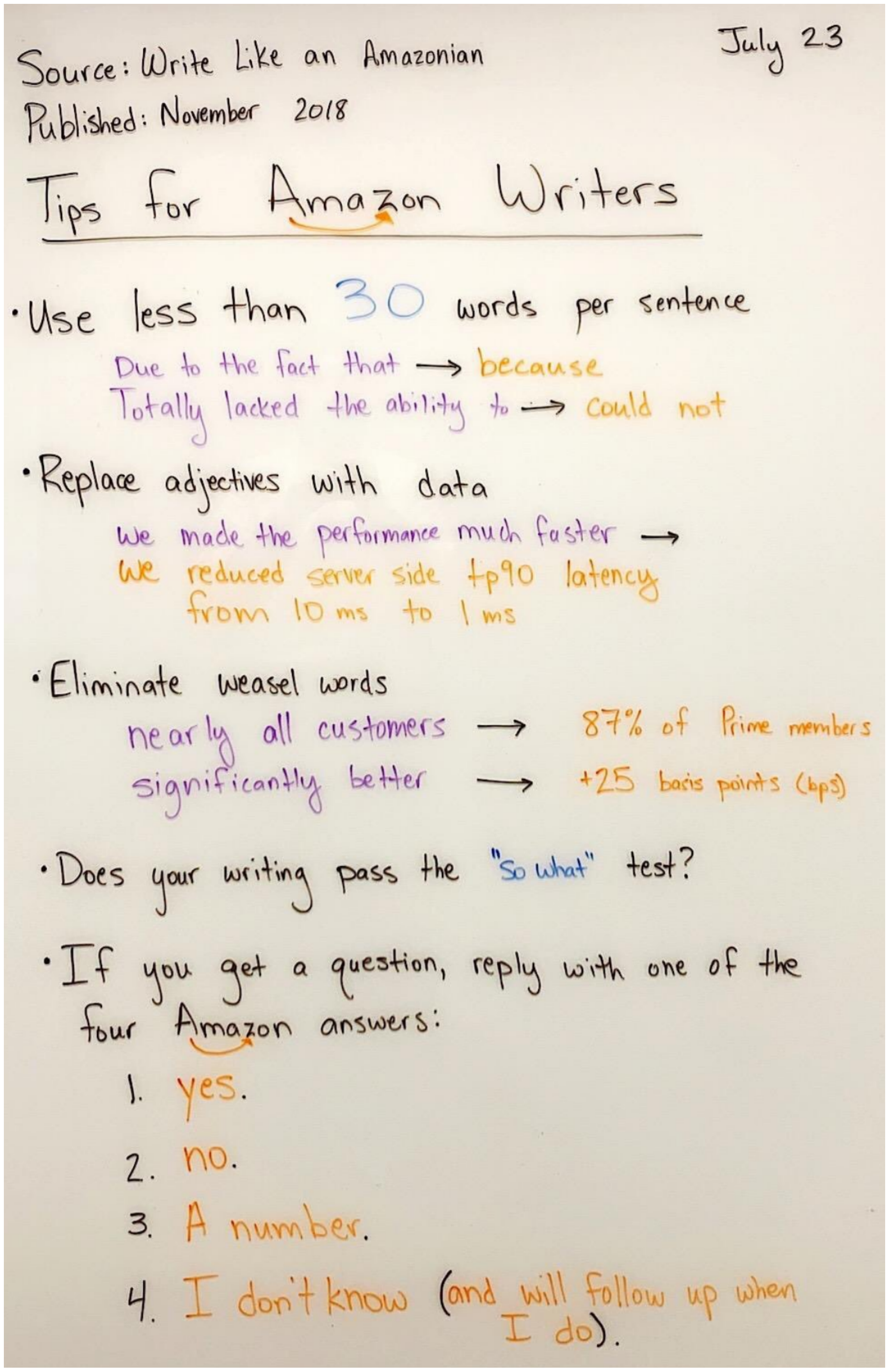
1. Prefix the subject with `[Action Required]` if you expect the reader to take action.
2. Structure your email in the following way (from 7 Tips for Better Executive Communication:
    1. First Paragraph — no more than 2 sentences. This would be the headline and critical information
    2. Second paragraph — 3–7 bullets. The more bullets, the lower line length. Aim to not wrap sentences.
    3. Third (final) paragraph — no more than 2–3 sentences focused on the action needed, follow up, and timelines.

Meetings

1. Avoid shitty brainstorm sessions.
    - Don't defer decisions to meetings. Make decisions on the spot, communicate it over long-form writing, and use the meeting to discuss it.
2. Encourage proposals to be written as Amazon-style "6 pagers" and "2 pagers".
3. Always end a meeting with actions, owners and timing, so it's clear what next steps are.
4. For staff meetings, go around the table and asking reports what their biggest concerns are.
    - Many managers want to attend executive staff meetings, as it makes them feel needed and puts them in the know. I made use of this desire by setting a price of admission to the meeting: you had to fess up to at least one thing that was 'on fire.'" - Horowitz

Hiring

- The best programmers are not marginally better than merely good ones. They are an order-of-magnitude better, measured by whatever standard:



Source: Write Like an Amazonian

Published: November 2018

July 23

Tips for Amazon Writers

- Use less than 30 words per sentence
  - Due to the fact that → because
  - Totally lacked the ability to → could not
- Replace adjectives with data
  - we made the performance much faster →
    we reduced server side tp90 latency from 10 ms to 1 ms
- Eliminate weasel words
  - nearly all customers → 87% of Prime members
  - significantly better → +25 basis points (bps)
- Does your writing pass the "So what" test?
- If you get a question, reply with one of the four Amazon answers:
  1. yes.
  2. no.
  3. A number.
  4. I don't know (and will follow up when I do).

Figure 4: Tips for Amazon Writers

conceptual creativity, speed, ingenuity of design, or problem-solving ability.
– Randall E. Stross

Hiring is the most important decision a manager makes.

What to look for in senior engineers:

1. **Owner**. Takes ownership of a problem even when it's not 100% their responsibility; understands the why.
    1. "Tell me about a time when you took on something significant outside of your area of responsibility. Why was it important? What was the outcome?"
    2. "Tell me about a time when you made a hard decision to sacrifice short term gain for a longer term goal."
2. **Handles ambiguity**
    1. "Can you tell me about a time when you had to solve an ambitious problem? Why was the problem important?"
    2. "Can you tell me about a time when you had to make a decision without complete information? What was the situation? What risks did you take? Why did you make the decision you made?"
3. **Team player**. Takes feedback well.
4. **Communicator**. Can articulate ideas at different levels.
    1. "Describe to me something you know well."
    2. "You mentioned X in your resume. Explain it to me as if I've never come across it?"
5. **Teacher**. Enjoys growing other engineers. Especially important for senior-level engineers.
    1. "Tell me of a time where you helped someone in your team grow."
6. **Deep diver**. Digs a level deeper to understand what's happening under the hood.
    1. "Tell me about a time you were trying to understand a problem on your team and you had to go down several layers to figure it out."
7. **Simplifier**. Simplifies problems instead of just hacks at things and adds tech debt. Does the person have a build vs. buy mentality.
    1. "Tell me a about a complex problem that you solved with a simple solution."
8. **Missionary**. Interested in company's mission or technology.
    1. "Explain to me what your current company does and why it's important."
    2. "What interests you working at [COMPANY]?"

What to watch out for:

1. **Short tenure at companies**. If a candidate typically stays at companies for less than a year, ask them why. There might be perfectly valid reasons or it might indicate a pattern that the person is difficult to work with.
    - Why did you only work at X for less than 1 year?
2. **Menu of technologies**. Resumes that just list technologies used instead

of problems solved may indicate person may not be thinking big picture. Also a typical trait of junior engineers.
3. **Long resumes**. More than 2 pages may indicate the person has difficulty distilling what's important. That being said, there can be culture reasons for long resumes. For example, in some European countries, resumes are encouraged to be long.

Onboarding

Having a good onboarding process is crucial to the success of your team and new team members. It ensures team members are contributing as early as possible and are assimilated into your processes and culture.

An onboarding process is successful if your new team member can contribute a bug fix on the first day of joining.

Onboarding material: 1. Team mission 1. How is your team moving the needle for the customer? 2. Team members 3. Repositories and services 4. Documentation: 1. Code contributing guidelines 2. Ticketing process. E.g. label and story pointing guidelines. 3. Glossary of terms. 4. Releasing code. 5. How tos (e.g. migrate database, add secrets) 5. Getting started: 1. Installation instructions (e.g. Docker, postgres). 2. Getting the right accesses (e.g. PagerDuty). 3. Running your apps locally. 6. Meeting setups. Who should your new team member meet with?

Announcing change

1. Examples of change: promotions, reshuffles, restructuring.
2. Acknowledge the difficulty or opportunity of the change.
3. Appeal to emotions by using narrative to explain the why.
    1. Why is this change important now for the company?
    2. Who does this change effect?
4. Appeal to logic by using facts.
    1. What metrics will this change achieve?
5. Socialize the change to get buy in.
    1. Start with the people who it affects the most.

Further reading

1. Amazon Leadership Principles
2. Manager's Path: Excellent guide for all levels of management.
3. 97 Things Every Engineering Manager Should Know: Collection of management tips from various practitioners.
4. The Pendulum or the Ladder: On the challenges of being a manager who wants to stay technical.

5. Hard Thing About Hard Things: More at the executive-level side but still a worthwhile.
6. What You Do Is Who You Are: Why company culture matters and how to establish one.
7. Engineering Management Repo: Great collection of management articles.
8. Hiring Engineering Leaders: Hiring engineering leaders.
9. How to Hire Smarter than the Market: Berkson's paradox and engineering hiring.
10. LifeLabs Learning. Great workshop for new and experienced managers. I learned a lot from it on feedback, coaching and 1-1s.
11. How Software Engineers Can Help Interview Their Future Managers: A list of questions for engineers to ask when interviewing managers.
12. Getting more from your one on ones: How to measure successful one on ones.
13. 7 Tips for Better Executive Communication: Tips on how to communicate effectively.
14. Toxic Meeting Culture: How to avoid meeting antipatterns.
15. The Feedback Fallacy: focus on replaying positive performance so team members know how to repeat excellence.