Table of Contents

Introduction

This site contains mainly links that I've collected over the years of maintaining A Fresh Cup. Hopefully you'll find some of it useful.

Feel free to contact me if you have suggestions or corrections.

General Management and Leadership Books

- Ask a Manager: A compendium of advice from Allison Green, author of the popular [[https://www.askamanager.org/) blog. It's in bite-sized Q&A format, covering a variety of things from how to deal with smelly co-workers to what to do if you throw up during a job interview to how you as a manager can coach people away from unrealistic goals. A good reference work that you can read and learn from in small doses. Even if you don't hit all of these exact situations (really, how many people have to deal with a bonkerballs co-worker putting curses on people?), Green's overall attitude of respect for everyone and honest conversation is a great example for us all. (Reviewed by Mike Gunderloy).
- The Advantage: Lencioni is pretty well known for his business fables (FIVE DYSFUNCTIONS OF A TEAM et al). This is his non-fable book, a synthesis of his work and the points he made in those other books to take an overall look at organizational health and its benefits. He walks through what he's found necessary about high-performing organizations, from leadership alignment to good hiring practices to which executive meetings are necessary and how they should be structured. I found the discussion of core values (and the other types) very useful, among other things. There is a fair amount of advertising for his consulting group sprinkled throughout, but it's more "we could help you with this" than "if you want the real secrets pay us" so it wasn't annoying; between the book and the handouts on the book's website, they appear to be giving away their whole system for free. Recommended. (Reviewed by Mike Gunderloy)
- The Best Team Wins: Heard the author on a podcast & ordered the book. I'm glad I did, even though a lot of the motivational techniques in here leave me cold. I'm an aging baby-boomer and so a lot of the things the authors have to say about those born later leave me confused. But they also match up with some of what I'm seeing at work. There is plenty of concrete advice here for motivating individual contributors and entire teams, including a long list (101!) of techniques to try. (Reviewed by Mike Gunderloy)
- Brave New Work: Dignan digs into what some organizations are doing to change the rules, then walks through a set of chapters on evaluating "the operating system" of your own organization and possible alternatives (this is the strongest part of the book in my opinion) and closes with a section on how to run a change process. Bear in mind that this is his own work, which means it's based on actual experiences in helping places change. I've got a bunch of sticky notes to review for the future as our company grows; the sections that ask penetrating questions on topics like strategy and resources, together with the guidance on what it means to be "people positive" and "complexity conscious" seem quite valuable to me. The main question in my mind is whether these good, innovative companies are destined to be trampled by the Facebooks and Ubers of the world, but we won't know that for another decade or more. (Reviewed by Mike Gunderloy)

- The Checklist Manifesto
- Crucial Conversations: Skills for getting past conversations that have turned into battles. The authors bring out just a few critical skills that are designed to get things back to the point where everyone involved is contributing to a pool of meaning. They won't always work, but they do help. The main challenge I have is to remember this stuff in the heat of the moment, but as they point out, anything that moves you in the right direction is worth doing. (Reviewed by Mike Gunderloy)
- The Daily Drucker: It took me a year to read this one, but after all, that's what it's intended to take. Bite-sized (no more than a page or two) chunks from Drucker's decades of writing on management, together with action points that can be used as springboards for thought. A lot of these didn't apply to my own circumstances as I read them, but enough things did, in fields from time management to marketing, to make it worth my while to exercise my brain on this book daily. (Reviewed by Mike Gunderloy)
- Deep Work: I'm pretty conflicted about this one. On the one hand, I do think the concept of deep work is valuable. The idea that some things require focus & concentration to achieve, and that you won't hit that focus and concentration by accident, is useful. But on the other hand, Newport's arguments in favor of it are largely a mix of anecdote, poorly-justified pop science of the Ted Talk variety, and fallacious logic. When he gets around to arguing that social media is worth quitting, though he himself has never been deeply involved, the whole takes on the flavor of a virgin producing straw-man arguments to justify avoiding sex. (Reviewed by Mike Gunderloy)
- Dynamic Reteaming: This one is currently in not-quite-final form at Leanpub, but although it still needs a tiny bit of copy editing, it's practically done. Helfand draws on her own experience in several growing tech companies, as well as interviews with other coaches, managers, and non-managers, to identify some patterns of change on teams. There's thinking here about team lifecycles, antipatterns to avoid, onboarding people to new teams, among other things. I especially liked the stories of companies that have had success allowing people to choose their own teams, either as part of a major reorganization or on a regular basis. (Reviewed by Mike Gunderloy)
- The Effective Executive
- The Effective Hiring Manager: I grabbed this one as soon as it was available and I'm glad I did; a lot of Horstman's advice will inform my future hiring process. That said, many many people are going to hate this book because it is bluntly about doing a good job of hiring based on job fit and merit. If you're one of the people who feels diversity & inclusion must be considered when hiring, well, those words aren't even in the index. There are also other times that you'll want to bring someone in based on factors other than proven ability in job skills (hiring interns, for example). But if you want a step-by-step process that leads to efficient decision-making and removes dependence on "gut feeling", and that goes from reading CVs straight through onboarding, you've come to the right place. (Reviewed by Mike Gunderloy)
- The Effective Manager
- An Elegant Puzzle: Will has been a senior leader at several large software companies including Stripe, Digg, and Uber. This is his collection of "how to

do it" recipes for success. I suspect it will have the strongest appeal to leaders who have come up from the software development ranks and who find people messy and uncomfortable. There is some material here on areas like hiring and culture, but the book is at its strongest when talking about things like the perfect n-step process to cold-recruit people based on their LinkedIn profiles, or how to size and reorganize teams. If you believe that management can be reduced to the same sort of science as databases, this is the book for you. I'll keep it on the shelf, but I don't think it'll be my go-to. (Reviewed by Mike Gunderloy)

- FIRE: How Fast, Inexpensive, Restrained, and Elegant Methods Ignite Innovation: An argument that better work comes out of projects that are deliberately "fast, inexpensive, restrained, and elegant". The author comes out of the engineering & military communities, where he's certainly had lots of opportunity to observe projects that miss all of those targets, soak up money, and deliver nothing of value. But the principles apply in software development as well. Constraining things can have a wonderful focusing effect. (Reviewed by Mike Gunderloy)
- The Five Dysfunctions of a Team:As it says on the tin, this is a "fable" and as such it can come out however the author wants. So, there's a happy ending, where the new CEO manages to build a high-performing team. Nobody wants to read a fable about the teams that crater. I'm not convinced Lencioni has identified THE way to high-performing teams, but a lot of this stuff strikes me as A way, and the connection between vulnerability and trust is, I think, more important than many small-team leaders give it credit for. So for reinforcement of that point, as well as for being a good story told well, I give it a five. I wonder, though, how many leaders and prospective leaders read this one through the warm glow of "the new CEO is so sharp, and her direct reports are such bozos...why that's exactly the way it is here!" Not Lencioni's point at all, but I'll bet it helps with sales. (Reviewed by Mike Gunderloy)
- Flight of the Buffalo: I read this one at roughly the same time as Pflaeging's ORGANIZE FOR COMPLEXITY, and it wasn't long before I got an eerie "separated at birth" notion. Those who think this agile decentralized stuff is all new might want to dip into FLIGHT OF THE BUFFALO, now 25 years old. The central theme: that you get better leadership by helping your people grow into a flock of self-organizing geese than you do by being the lead buffalo who does everything (and thereby becomes the indispensable bottleneck to everything in the organization). An easy, though somewhat long read, it teaches a series of lessons about why and how a good leader will let go, and how to challenge your employees to succeed by delivering great performance for customers. (Reviewed by Mike Gunderloy)
- Get Smart!: How to Think, Decide, Act, and Get Better Results in Everything You Do
- The Goal: A Process of Ongoing Improvement: This one gets super reviews but I'm at best lukewarm about it. It's an early example of the "teach business principles through a novel" genre. In this case, the bumbling plant manager gets whacked over the head by his Israeli physicist friend until he figures out that flow through his plant is more important that keeping everyone busy. You'd think that the piles of inventory choking things off would have been a

clue. There are things to learn here, but they could be taught in half the size, without the bad romantic subplot or the amazingly bad portrayal of Boy Scouts. (Reviewed by Mike Gunderloy)

- Grinding It Out
- How F*cked Up Is Your Management?: You can read the title two ways: "how fucked up are these idiots I report to" or "how fucked up is my own management style". I think the second reading is much more productive. From the editors of The Co-Pour (https://mfbt.ca/), (and a lot of the content started there), this book takes a hard look at some common management and geek ideas and rethinks them. Think you're doing enough for diversity? Think your unlimited vacation policy is a benefit? Think again. (Reviewed by Mike Gunderloy)
- How to be an Inclusive Leader: An excellent guide to those of us trying to navigate the tricky and confusing waters of diversity & inclusion in the workplace. Brown draws on her large experience in the field to provide a framework for thinking about where you are on any particular dimension of diversity, from unaware through aware and active to advocate. She also makes the point very strongly that this is not a simple linear journey, and that you should be prepared to move back and start over more than once. I expect I'll be reading this one again in the future, both for the framework and for the concrete ideas about what to do if you really care about these areas of life. (Reviewed by Mike Gunderloy)
- How to be Happy at Work: I ran across this one thanks to the How To Be Awesome At Your Job podcast, and picked up a copy. It's a lot more academic than I had expected: even though this is aimed at a popular audience, McKee has plenty of research background and she pulls in a lot of work she's been involved with in the past, notably on emotional intelligence. There's no magic spell to be happy, of course, but there are techniques that can help in many circumstances, and she makes a nice organized presentation of them. I personally got the most value out of the discussion of hope at work and the value of having friends, even though I'm not one to easily make friends. It's nice to see an empowering set of ideas in one place, though I'm not 100% sure I'll have luck putting them into practice to change things outside myself. But one of the messages here is that changing yourself is a good starting point. (Reviewed by Mike Gunderloy)
- Influencing Virtual Teams
- The Innovator's Hypothesis: "How cheap experiments are worth more than good ideas" is the subtitle here, but be warned: if you're in a small startup, you may not have the same definition of "cheap." Schrage has clearly worked with a bunch of big companies, and his idea of a radically streamlined process is to give a team of 5 people 5 days to come up with a portfolio of 5 experiments that take no more than 5 weeks and $5000 each to run. But even if that's too rich for your small-scale blood, the basic notion that actually getting out and DOING things trumps just THINKING about them is worth internalizing. And who knows, you might even pick up some brainstorming ideas that you can apply to your own experimental practice. (Reviewed by Mike Gunderloy)

- Lead Inside the Box: A framework for thinking about leadership in organizations based on a matrix that balances leadership capital invested vs. results obtained. I'm not currently leading a team, but this one was valuable to me from the other side of the table: I did some deep thinking about which of my behaviors at work are causing my boss to invest unwarranted amounts of his leadership capital. I came away resolved to change a few things, and that's about all I can expect of any book. (Reviewed by Mike Gunderloy)
- Lead Right for Your Company's Type
- Lead Yourself First: An excellent book focused on the importance of solitude to leaders - though that could just as well have been cast as "the importance of finding time to focus." While many leaders get that time through being away from everything (running, vacationing, or just closing the office door), others may be able to focus just by a change of setting away from the everyday noise. I'll be chewing on this one for a while, especially as I have fallen deeply into the trap of instant response to everything myself. (Reviewed by Mike Gunderloy)
- Leaders Eat Last: The basic idea here is simple, clear, and worth knowing: one of the prime jobs of a leader is to encourage a "Circle of Safety" in which every employee can do their best. Unfortunately that simple idea isn't up to carrying a 300-page book. There's a great deal of pop biochemistry and potted history and references to things like the Milgram studies (which have since had their moment in the revisionist glare) thrown in, none of which I found especially illuminating. The whole thing strikes me very much like an extended TED talk; you have the feel while listening that you're learning Important Things, but when it's all over, there isn't a whole lot to hang on to. (Reviewed by Mike Gunderloy)
- The Leaders Pocket Guide
- Make Your Bed
- The Making of a Manager: A nice addition to the shelf of management books aimed mostly at the software industry (though certainly some of the techniques here will work in other industries as well). Julie Zhou is currently a VP at Facebook, but she still remembers what it was like to be a new & clueless manager when the company was much smaller. Her target audience is precisely those who have suddenly found themselves in management. The sections on dealing with your first three months are great, and the sections on good meetings, good hiring, and good O3s all stand out as well. Recommended if you're early on the management track, or thinking about heading in that direction & wondering what you're in for. (Reviewed by Mike Gunderloy)
- A Man for All Markets
- Management 3.0
- The Manager's Path: This one traces a typical career progression for software engineers, from individual contributor to mentor and team lead, and then on into successively higher levels of management ending up in the C-suite. The author has made that journey herself, and offers advice based on her own career -- which, you ought to keep in mind, is just one data point, albeit an extended & successful one. This book was most valuable to me in terms of seeing what I didn't want to do in the future. It's worth reading at

least as far as you aspire to go, and then skip forward to the last chapter on team culture. (Reviewed by Mike Gunderloy)

- Managing in the Gray
- The Mom Test: Really helpful tool kit for customer learning conversation (reviewed by Daniel Baily)
- No Hard Feelings: A leadership book that stepped out of my usual comfort zone and made me think, which is about all I can ask. The authors dig into the uses (and some of the misuses) of emotion at work, arguing that in the modern workplace selective vulnerability is a necessity. I'll likely come back to this one in the future when I hit some situation that I can't quite work through on my own. (Reviewed by Mike Gunderloy)
- Once an Eagle: At 1200+ pages, reading this one was a serious commitment - but worth it. The story of Sam Damon, professional soldier from WWI to the VietNam era (and secondarily of Courtney Massengale), it tries to show by example what leadership looks like, as opposed to simply executive ladder-climbing. The portrayal of the necessity of a moral backbone, and care for one's subordinates, is well-done. No wonder they're still reading this one at American's military academies. (Reviewed by Mike Gunderloy)
- One Mission: This one was an inspiring read as I think about our company's management structure and how it might get through a round of growing pains. It's a follow-up to Team of Teams, continuing to dig into how lessons from the Special Operations Task Force run by General Stanley McChrystal can be applied to the fast pace of today's business. Fussell makes a case -- I think a convincing one -- that no matter how good your managers are the "solid-line hierarchy" of an organization isn't enough to keep up any more. The best managers act as information pumps, collating information as it moves up the organization and sending it back down to other subordinates who need to know. But this is an inherently slow process, and one subject to communications breakdowns as messages get passed multiple times (remember your childhood game of "telephone."). On the other hand, a purely decentralized organization with no managers, a la holocracy, is incapable of large-scale coordination (this is my interpretation, so blame me rather than Fussell if you're a holocracy fan). The solution proposed in this book is to overlay a network of "dotted-line relationships" between small teams, recognizing that there are individuals in the organization who are well-connected. That's not new. What is new is seeing management's task as making sure those individuals communicate regularly, that the entire organization is included, and that key connectors are empowered to decide things even at the expense of breaking the organization's rules. There's a lot more here, and I find it all pretty convincing. I'll be recommending this one for a while, I'm sure. (Reviewed by Mike Gunderloy)
- Only the Paranoid Survive: This is Grove's book-length argument that one of the main features of successful upper management is to recognize "10" factors that lead to strategic inflection points. Recognize and seize these opportunities, and your company can thrive; miss them at your peril. Overall, it's a pretty good read, and gave me a way to think about my position in my current company (I'm one of the Cassandras). I wish that there was more concrete advice on telling signal from noise, but that's part of the challenge:

figuring out which factors matter, and which are immaterial. There's a final chapter on treating your own career as a mini-enterprise which I found pretty uncompelling (it has the feel of something tacked on later to enhance sales), but overall I'm glad I read this. (Reviewed by Mike Gunderloy)

- The Optimistic Workplace: There are a bunch of themes intertwined here, on the way to a better workplace. There's an argument that management is better replaced by stewardship; there's a notion that work need not be a soul-sucking trade of time for money; there's evidence that optimism encourages people to perform better. There's also a 90-day plan for stewards who would like to implement these ideas in their own workplaces. While I don't expect to be personally executing on that plan any time soon, I did appreciate the reminders (and evidence) that things improve on many fronts if you act like you're dealing with adults who want to do their best, instead of getting sucked into an endless cycle of politics and distrust at work. (Reviewed by Mike Gunderloy)
- The Power of Full Engagement: The central argument here is that if you're busy and focusing all your attention on time management, you're doing it wrong. The authors argue that energy management, specifically various analogs of interval training that allow you to push yourself and then recover, is even more important. The argument is buttressed by a bunch of anecdote and case studies where they were able to successfully turn lives around, though it's unclear to what extent the stories they tell are composites rather than actual individual successes. I generally like the central idea, but I dislike the attempt to put everything into a neat crystalline structure, where what is said about physical energy is perfectly echoed in discussing emotional, mental, and spiritual energy. In my experience, life isn't that pretty. Much of the book also feels like padding or advertisement for their services. But the notion of building rituals to recharge yourself is, I think, an important one, and if you've never stumbled across it on your own that makes this book at least worth skimming. (Reviewed by Mike Gunderloy)
- Principles: I put this one in the category of "total systems." From the inside, Ray Dalio has written down, in a logical and ordered fashion, all the things that he thinks contributed to his own success and that of Bridgewater Associates. Whatever else you say about this, it's objectively true that he's made a boatload of money following this recipe. (The book also contains a biographical section as background, but that's pretty dull and self-congratulatory). His principles run from "Embrace Reality and Deal With It" to "Don't Overlook Governance" and give a set of tools for life & work. He comes to the conclusion that an idea-based meritocracy is objectively the best way to organize a workplace, and it's very clear that he's used these tools to select people who agree with him to turn Bridgewater into a machine (his word) that functions very well in the world of finance. If you accept his premises, then Dalio's principles explain everything. This I think is the prime characteristic of a total system: from the inside, it is complete, an explanation of the entire world, and you need nothing else. The problem comes when you realize that there are many conflicting total systems out there: Ayn Rand, L. Ron Hubbard, Karl Marx, and Malaclypse The Younger are among others who have produced complete explanations for the world. They can't all be

right at the same time. I'm sure there are things in here that will influence me moving forward; some of his ideas on exposing and settling disparate points of view, for example, seem useful (though the description of meetings at Bridgewater, with real-time "dots" micro-feedback from participants, is frankly horrifying). But I can't imagine buying into the premises to the point where I'd be happy in an organization run 100% by Dalio's Principles. (Reviewed by Mike Gunderloy)

- Quality is Free
- Radical Candor
- Resilient Management: A short (just over 100 pages) book for new engineering managers. I know many managers who should read this. Unfortunately most of them probably won't, because Lara focuses on all the squishy stuff about building trust and helping your team and getting past the forming and norming stages, rather than on cookbook recipes for managing projects and figuring out velocity. If you're managing people rather than robots, and especially if you're not already familiar with things like the BICEPS model, or the difference between coaching, sponsoring, and mentoring, I can't recommend this one highly enough. (Reviewed by Mike Gunderloy)
- Start With Why
- Strategic IQ: Wells pulls together a lot of different threads here to try to examine the idea that some firms are better at identifying and executing profitable strategies than others - or, in his language, they have "high strategic IQ." His taxonomy of approaches and pitfalls led me to reflect that I think I've seen every single one of the dysfunctional low IQ behaviors he identifies during my career -- and I'm not sure I've ever worked at a firm that would make his cut for "high." After looking at the idea of strategic intelligence, he discusses ways that firms can improve their structures and their people to compete better. It doesn't all quite come together into a coherent whole for me, but there are plenty of thought-provoking ideas: more than enough to justify buying a copy. (Reviewed by Mike Gunderloy)
- Switch: How to Change Things When Change is Hard: A lot of people find this inspirational; I felt it was at best ho-hum. The Heaths provide a reasonable framework for change management, though far from the only possible framework, and if you need to organize your thinking their Rider-Elephant-Path paradigm may be of help. But they are masterful at jumping from particular just-so stories to sweeping principles, and I'm past appreciating that style. One to read once, but probably never again. (Reviewed by Mike Gunderloy)
- Thinking in Bets - I didn't have too high a hope for this one: a book on how to make decisions written by a professional poker player. I expected some variation of "know your opponents and learn how to bluff." Well, I was wrong, and also pleasantly surprised. Annie Duke has a lot of things to say about making better decisions, chiefly by trying to avoid some common mistakes (like depending on hindsight or wishful thinking) with the help of friends. She pulls together everything from John Stuart Mill to the poker table to make these points, and it's an engaging read with a lot to think about in the end. (Reviewed by Mike Gunderloy)

- The Toilet Paper Entrepreneur
- Why Leadership Sucks
- Your Oxygen Mask First: This one is from an executive coach, and largely directed at executives so busy that they need a short, no-nonsense checklist of things that they should be doing (or that they should stop doing). Despite that, Lawrence is big on making sure you have a life outside of work, as well as at driving hard while you're working. I'll likely read selected chapters a second time, paying attention to the end-of-chapter exercises as I do so. (Reviewed by Mike Gunderloy)

Agile, Scrum, XP, and Lean Books

- Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations: This book tries to take a rigorous, data-driven approach to teasing out the impact of DevOps practices on organizational productivity. The authors come to conclusions like "lean product development causes less burnout" and offer their survey-based research as evidence for their conclusions. I desperately *want* to believe this book, because their conclusions accord with my own prejudices: that lean, agile, and transformational practices contribute to organizational improvement. Unfortunately, I just can't quite close that gap. One of the telling points for me comes in Chapter 15, where the authors say "We did not collect data from professionals and organizations who were not familiar with things like configuration management, infrastructure-as-code, and continuous integration. By not collecting data on this group, we miss a cohort that are likely performing even worse than our low performers." I don't see any evidence for that assumption, or any reason to believe that their self-selected snowball sample is representative enough to tell us anything about the actual impact of various practices on software development. Nonetheless, I'm happy to see this book, because it pushes things in the direction that I'd like to see them go. Whatever my own concerns about methodology, you may be sure I'll be citing this in my attempts to get my own upper management to move to more "agile" ways of working, simply because I find them more pleasant. (Reviewed by Mike Gunderloy)
- Agile Retrospectives: This one has been on my shelf for a decade, but I just pulled it off and read it again since I'm finally part of a team that is serious about retrospectives. Even though I'm not the agile coach for the team, I found a lot to think about here - there are many, many retrospective techniques that can be used, and I'll be interested to see us try some of them. This was also the first time in a long time I've thought about release or project retros, instead of just sprint retros. (Reviewed by Mike Gunderloy)
- Create Your Successful Agile Project
- Extreme Programming Explained
- FIRE: How Fast, Inexpensive, Restrained, and Elegant Methods Ignite Innovation: An argument that better work comes out of projects that are deliberately "fast, inexpensive, restrained, and elegant". The author comes out of the engineering & military communities, where he's certainly had lots of opportunity to observe projects that miss all of those targets, soak up money, and deliver nothing of value. But the principles apply in software development as well. Constraining things can have a wonderful focusing effect. (Reviewed by Mike Gunderloy)
- Lean Software Development
- The Lean Startup: I ran across this one on my shelf, having completely forgotten I'd picked up a copy and never read it. So I went ahead and did so. I can go along with crediting this book with doing a lot to promote "lean" thinking in the context of software startups, which makes it an important

current in agile thought. But I couldn't help noticing how many of the companies he cites in this 2011 book have failed to make an actual splash. That's a common failing of business books, of course - but it does make having faith in conclusions a bit harder. On the plus side, the notion of knowing what you're optimizing and experimenting fast is a good and useful one. On the minus side, I suspect we would all have been better off if "pivot" as a concept had never been introduced to startup-land. It's not Ries' fault, but that's ended up as shorthand for "we haven't spent all your money yet, let's try the next stupid idea." (Reviewed by Mike Gunderloy)

- Making Work Visible: Subtitled "Exposing Time Theft to Optimize Work & Flow," this is the handbook of kanban practice that you didn't know you needed. Degrandis does a great job of showing why kanban is a practical technique, not just a different set of ceremonies, by motivating it around the need to find (and eliminate) the things that waste time at work. It goes through a lot of different ways to tweak a kanban process for specific purposes, and it's likely that any struggling team will find some immediate relief here. (Reviewed by Mike Gunderloy)
- The Openspace Agility Handbook: A little bird (at a previous job) clued me in that this one was getting read in our C-suite. Naturally I had to grab a copy for myself. It's a quick read, and one that proposes a structured course for launching an agile transformation (though much of the structure is concerned with opening up space to avoid existing structures). A few observations: 1) The authors (and developer of OpenSpace Agility) have a bad case of Too Many Proper Nouns With Capital Letters Syndrome. The editor in me wants to make them rewrite to conform to standard rules of English. 2) It was worth the work to get past the T.M.P.N.W.C.L.S issues and engage with the actual material. The path the authors propose for launching a successful agile transformation is at least plausible. 3) Even with this structure, it's all going to come down to people & commitment. It seems pretty clear that the C-suite could launch an empty OpenSpace Agility exercise, collect the results, pat themselves on the back, and go on with business as usual. It would be pretty easy to only extend invitations to participate to "safe" employees, i.e., those who will not rock the boat. 4) I suspect the choice of Facilitator is going to be absolutely critical. I think it will need to be someone with respect across the organization and either (a) sufficient political capital to make upper management listen or (b) sufficient IDGAF to ignore upper management. 5) There are some novel ideas here, including inviting even those who are opposed to an agile transformation, and casting the whole process as a game. I don't know if they'll work, but when existing ideas aren't working it seems to me that any new idea is worth trying. 6) Like most agile books directed at a context wider than software development, this one tries to navigate between a "one size fits all" prescription and a "do whatever you want" description. The touchstone here is whether practices conform to the Agile Manifesto. I'm not convinced this works but I don't have a better idea. 7) "Emergent leadership" appears to me to be an idea ripe for abuse by people who are determined to get ahead at the expense of others because they are more outgoing, vocal, or simply loud. I don't see any attention here to encouraging safe space for diverse voices, or to approaches other than the

"marketplace of ideas," and that concerns me. 8) There's an assumption here that you bring in an outside coach to help. That smells like a good idea to me, and any organization that's not willing to spend the money on an outside coach may not really be serious about transformation. 9) If the company is really serious about pursuing this path: (a) there will be quick movement and a sense of urgency about starting the process (b) the overall Sponsor will be at the very top level of the company, not a few levels down because "it's their job" (c) all parts of the company will be involved, not just engineering (d) travel and other costs will be covered for employees no matter where in the world they live. There are, I think, a lot of paths to agile. This one certainly looks plausible, and its structure may make it less scary to upper management than a complete "trust us and turn us loose" approach. I'd be happy to see my employers go down this road. And, if the rumor about them reading this book is true and nothing happens in 30-60 days, I'll be drawing some negative conclusions from that as well. (Reviewed by Mike Gunderloy)

- Organize for Complexity: Pflaeging presents his core ideas in this graphic-heavy book (suitable for both right-brained and left-brained readers): that "management" is a problem rather than a solution, that leadership is distinct from management, and that there is a difference between "alpha" (managed) and "beta" (led) organizations. Lots of connections here with things like agile, though mostly implied rather than spelled out. Radical decentralization of decision-making is one of his key themes, and he argues that this can lead to enormous competitive benefits. Best part: there is some prescriptive material on how to think about changing an existing organization into a beta form. Worst part: I think some of his history is a bit glib and misses the complexities of how we got current management structures, though it makes for a good story. (Reviewed by Mike Gunderloy)
- Project Myopia: Mostly a polemic about trying to mix traditional project management with agile software development. For people stuck in a project mindset, it might be an eye-opener. Briefly, the fixed scope fixed end attitude of traditional projects doesn't mesh well with the notion of delivering highest value first. Kelly brings a lot of arguments to bear on this, just in case your boss is reluctant to understand or you're a little slow yourself. (Reviewed by Mike Gunderloy)
- Scrum 101: A quick read about the basics of Scrum (100'ish pages most half blank). It's set up as a set of questions and answers, walking from the basics of the Agile Manifesto through the ceremonies and artifacts of Scrum. I have the same problems with it that I do with most Scrum books, which is that some people seem to mistake form for function, but as a reference it isn't bad. (Reviewed by Mike Gunderloy)
- Scrum - A Pocket Guide: Good and fairly short (100 pages) overview of scrum. One way they made it "pocket" though was by reducing the type to something like 7 point so it's eyestrain city for us older folks. Things I found interesting; 1. Presenting scrum as a game, with rules and tactics 2. The de-emphasis on "do it this one way" in favor of "here's the principle, there are many practices" 3. The growing movement towards "scrum and ... " as it gets into ever-larger and more diverse organizations. Things that annoyed me: 1. Tiny print 2. The self-congratulatory tone 3. The lack of editing by a native

English speaker All in all, it'll be on my shelf, close at hand, as long as I'm in an organization using scrum. (Reviewed by Mike Gunderloy)

- Scrum Mastery: This one has me thinking, which is an excellent recommendation. The author applies experience as a scrum master to identifying behaviors that move scrum masters from "good" to "great." When I look around my own company, I don't see a lot of these great behaviors - but I think that's due to lack or organizational support. So then the question is: what can the organization in general, and I in particular, do to help remedy that? (Reviewed by Mike Gunderloy)
- User Story Mapping: If you think user stories are just "As a (blank) I want to (blank) in order to (blank)" Mad Libs, then you've got a lot to learn here. Patton has a broad overview of stories as conversations, and he spends 250+ pages getting that overview to the reader. Covers everything from writing good stories, to the different sizes and uses of stories, to a variety of ways to hold workshops for different purposes, all revolving around the continuous conversation between all stakeholders. This one was an eye-opener; our company is agile, but there's a heck of a lot more we could be doing here. (Reviewed by Mike Gunderloy)

Software Management

- After the Gold Rush
- Deathmarch
- Debugging the Development Process
- Escaping the Build Trap - A solid read for anyone who aspires to be a product manager, works with product managers and wants to understand what they do, or who wants to drive their organization towards being product-led. In other words, pretty much every developer and developer-adjacent employee I've ever know. Melissa Perri does a great job of explaining why turning into a feature factory is a trap, and how to escape by focusing on real value delivery driven by hypothesis and experiment. The book also helps distinguish between project management and product management, and shows why the latter is a critical missing skill in many agile companies. Particularly relevant if you've struggled with deciding which things to deliver, or if you've been handed a veneer of "lean" thinking without any organizational change to philosophy to back it up. (Reviewed by Mike Gunderloy)
- Facts and Fallacies of Software Engineering
- Managing the Unmanageable
- Memoirs of a Software Team Leader
- The Mythical Man-Month
- Peopleware
- Rapid Development
- Ship It!
- Software by Numbers
- Software for Your Head
- Software Project Survival Guide
- Writing Solid Code

Architecture

- 37 Things One Architect Knows About IT Transformation: I read this & quite enjoyed it, in part because Hohpe is an engaging writer & he picks on big organizations a lot. But note the "About IT Transformation" in the title. Much of the material here is about how to get big old-economy companies to adopt more agile methods of work, and only tangentially about architecture – except inasmuch as he views the Chief Architect as a key agent of change. (Reviewed by Mike Gunderloy)
- Building Evolutionary Architectures: There are a lot of good points here about how to build large software systems that can withstand business change - though more of the points seem to be about what NOT to do. The actionable advice is a somewhat smaller piece of the book. The best part is the reminders that monoliths are not always bad; there are tradeoffs to be made. Ford has sweeping experience across many software trends, and it's nice to see microservices placed in the broader context. (Reviewed by Mike Gunderloy)
- Building Maintainable Software
- Design It!: I enjoyed this introduction to software architecture, while recognizing that it's clearly one architect's view and that there are a lot of conflicting definitions in the field. This one at least was refreshingly close to the software design activities I already know from a career of programming, rather than being an "enterprisey" muddle of vague generalities. (Reviewed by Mike Gunderloy)
- Guerilla Capacity Planning: An introduction to capacity planning in a land of fast change and imperfect information - though note that for Gunther "fast" is a 3-month planning horizon, so it's hard to integrate this into the agile world. Focuses mainly on basic scalability laws, the difference between the parallelizable portion of the work and the rest, and shows applications in a variety of areas from program decomposition to internet scaling. Lots of math interspersed with smartass comments, the latter helpfully collected in an appendix so you don't have to wade through the math. (Reviewed by Mike Gunderloy)
- Just Enough Software Architecture: A good introduction to the practice of software architecture for any reasonably-seasoned developer. The key takeaway here, and a point that Fairbanks hammers on early and often, is that the amount of modeling you should do is proportional to risk. There are no points for endless reams of documentation that no one ever reads in pursuit of "completeness." He uses mostly (the simpler parts of) UML to demonstrate basic modeling concepts, and provides plenty of standard tools to get started. I don't know a better text for the budding architect. (Reviewed by Mike Gunderloy)
- Patterns of Enterprise Application Architecture
- Software Systems Architecture: A solid textbook for developers already somewhat familiar with architectural thinking. It takes a views, viewpoints, and perspectives approach. While it does have an extensive catalog of the

most important artifacts in each of those categories, it also has extensive sections on the process of being an architect, from coming up with the business goals and engaging stakeholders through creating and validating the architecture. Somewhat heavy sledding in spots and not really concerned with the "agile" world but a valuable reference. (Reviewed by Mike Gunderloy)

- Software Architecture for Developers: A developer-centric view of architecture that spends quite a bit of space exploring the tension between agile and BDUF approaches. Personally I'm a fan of more up-front design than many agile teams, which may just mean I'm getting old. In the end I appreciated the multiple perspectives on the multi-faceted architect's role here. Maybe I still have enough career time left to be an architect when I grow up. (Reviewed by Mike Gunderloy)

Design Books

- Antipatterns
- Artful Design: This one was well out of my comfort zone, and I'm extremely glad I read it. Set up as a graphic novel with hundreds of photos, it traces the authors thoughts about design, humanity, engineering, purpose, and similar huge topics. With examples drawn from compute rmusic, games, and even social networks, it takes a very modern look at an old topic. I don't do a lot of designing but I expect I'll reach for the list of broad principles in this book the next time that I require inspiration. (Reviewed by Mike Gunderloy)
- Design Patterns
- An Introduction to Data Structures With Applications
- Mental Models: Aligning Design Strategy with Human Behavior: I'm glad I read it, and I'll probably never re-read it. Super-structured and prescriptive approach to understanding user needs and then matching them back to product features in order to build better products. At least I've got more understanding of what it is that information architects do. (Reviewed by Mike Gunderloy)
- The Mom Test: It's easy to get people to tell you that your startup idea is good: just make it clear that's what you want to hear, and don't require them to make any actual commitment ("Sounds like something I'd buy, great!"). It's a lot harder to get useful feedback, or to figure out what you can build that will actually sell. In this short (126 pages) book, experienced tech entrepreneur Rob Fitzpatrick shares what he's learned over the years that can help in this effort. The main lesson is to talk about their life, rather than your idea, but there's plenty more to dig into. Recommended to anyone chasing that elusive "product-market fit." (Reviewed by Mike Gunderloy)
- Product Roadmaps Relaunched: I'm not a Product Manager - which is to say, that's never been my title on a business card (not that I've had business cards for twenty years or more, but that's another story). But as with many other fields in a small tech company, I end up pinch-hitting in this area. From that point of view, this is an excellent book, focused on balancing a lot of competing factors: predictability vs. uncertainty, multiple stakeholders vs. each other, sales bs. development and so on. There are plenty of useful examples, but more than anything else, this is about having the right mindset and skills to help develop an overall plan, present it, and get buy-in from everyone concerned. (Reviewed by Mike Gunderloy)
- Ruined by Design: Those who ding this work because it's an unabashedly political rant and call to arms are, I think, completely missing the point. Many designers (and by that term Monteiro and I both include everyone who designs things, including software developers and product managers as well as UI/UX professionals) have become utterly complicit in the unethical practices of the venture-capital fueled giant tech companies, and it's time for this to just stop. If you don't agree with that statement, you might as well not read this book. But if you do agree (and I happen to be on that side of the argument), this is a fairly short summary of the mess we're in, together with a

few thoughts on how we might get out of it. I've certainly been guilty of designing bad things in the past - my stint at a combined multi-level marketing/adware company springs quickly to mind - but I've finally reached a point (either in my maturity or my career) where I have deliberately turned my back on that path. At this point I rate working on software that actually makes the world a better place far above squeezing money out of bigger fools. I hope that becomes a trend, though I fear that it will not. (Reviewed by Mike Gunderloy)

Microservices Books

- Building Microservices: A broad overview of Microservice design. As with other broad overviews (such as Susan Fowler's book) it does not get into the weeds with advice for any particular programming language or framework. Rather, you should treat it as a list of things to think about and patterns that have been proven in practice. The field is fairly young, though, so don't assume it's an exhaustive or authoritative list. (Reviewed by Mike Gunderloy)
- Practical Microservices: Build Event-Driven Architectures with Event Sourcing and CQRS
- Programming Models for Distributed Computation - "This is a book about the programming constructs we use to build distributed systems. These range from the small, RPC, futures, actors, to the large; systems built up of these components like MapReduce and Spark. We explore issues and concerns central to distributed systems like consistency, availability, and fault tolerance, from the lens of the programming models and frameworks that the programmer uses to build these systems."
- Production-Ready Microservices: There's a fair amount of high-level "mom & apple pie" guidance here on microservices, and a paucity of low-level implementation decisions. To be fair, a book trying to tackle low-level issues in every possible microservices language would be so large as to be unfinishable. The end-of-chapter and end-of-book checklists of things to think about are valuable as a survey of issues to tackle in your own march to a brave new microservices world. (Reviewed by Mike Gunderloy)
- The Tao of Microservices: I've read a bunch of microservices books in the last six months, and this is the best one I've found so far. I know that because it makes me want to go back, tear up everything we've done, and start over. Rodger offers a lot of advice down in the weeds about everything from deployment strategies to office politics to monitoring patterns, but the key insights here revolve around how to inspect requirements, decompose them into messages and services, and implement them. Thoroughly recommended. (Reviewed by Mike Gunderloy)

Operations Books

- Effective Devops
- A Practical Guide to Testing in Devops: An excellent read and reference for any organization concerned with shipping quality software in an evolving devops environment. There are great ideas here on how the role of testers and testing might change, how to socialize those changes (and to get buy-in from across the organization), and many links to articles, case studies, and tools that should get you inspired. Recommended. (Reviewed by Mike Gunderloy)
- Site Reliability Engineering
- Terraform: Up and Running

Security Books

- Threat Modeling
- Writing Secure Code

The Manager Track

A few links for anyone thinking of hopping to the management track:

Start Here

- The Manager's Path (book) - This is the number one best reference I know for technical individual contributors moving to management

Books

- The Making of a Manager: What to do When Everyone Looks to You - From Julie Zhou, who went from individual contributor to Facebook VP.

Getting Started

- Making the Most of Your First Three Months as a New Manager - Excerpt from **The Making of a Manager**.
- Advice for first-time managers
- A Reading List for New Engineering Managers
- How to fail as a new engineering manager
- Engineering Management: Lessons learned in first year
- Becoming a bad-ass engineering leader: 5 tried and true lessons from a woman of color
- The Seven Areas Of Software Management
- Everybody's Crazy: Why Management is Hard (And What to Do About It)
- Flavors of Engineering Management
- How to Lead
- What is an Engineering Manager?

Career Paths

- Engineering Management: The Pendulum Or The Ladder - Not sure whether you want to head for management? Read this first.
- Engineering Growth Framework - Career paths from Medium
- Progressing from tech to leadership - things to think about as you move into a leadership role
- Switching from Engineering to Management
- I Didn't Want to Be a Manager Anymore—and the World Didn't End - Going back to a technical role
- On being an Engineering Manager - Another tale of trying management and pulling back
- Software Roles and Titles
- Career Growth Frameworks in Software Engineering: A Review

Coaching, Mentoring, and O3s

- Coaching
- Keeping your 1 to 1s fresh - How to not just turn them into status updates
- Ask "How's It Going?"
- Engineering Management Distilled: A guide to one-on-ones
- How to Deliver Constructive Feedback in Difficult Situations
- The Feedback Fallacy
- How great managers give and receive feedback
- 10 Mistakes You Should Avoid During Your One on One Meetings
- The one-on-one meeting template for your end of the year review
- Mentored by Wizards, Part 1 – “You’re Worthwhile”
- Coaching tool: The ORG team building model
- Mentoring Developers: Best Practices From Uber - Interview With Gergely Orosz
- One-on-ones are my most valuable meetings; here’s how I run them
- Essential Meetings to Have With Your People as a Manager
- 1 on 1 Meeting Questions - An extensive crowdsourced list.

Communicating

- The Senior Engineer’s Guide to Helping Others Make Decisions
- Management bugs: 18 months later
- How transparent should you be as a leader?
- The five types of communication problems that destroy company morale
- The Secret to Being a Better Boss: Create a “How to Work With Me” Manual
- BLUF: The Military Standard That Can Make Your Writing More Powerful
- How (some) good corporate engineering blogs are written

Ladders, Evaluations, Promotions and Related Topics

- A software engineering manager guide to measuring a software engineers performance
- Square’s Growth Framework for Engineers and Engineering Managers
- Stripe’s Will Larson on Designing a Performance Management System from Scratch
- Multi-team Software Delivery Assessment
- Engineering Career Development at Etsy
- Etsy Engineering Career Ladder
- Job Titles & Levels: What Every Software Engineer Needs to Know
- Individual Performance Appraisals, Just Say No! - If the team does the work, why are you grading the people?
- How I Do (Hopefully) Fair Performance Reviews for Software Developers

On-Call/Incident Management

- When Does an Investigation End? - Thinking about incident follow-up.
- Making On-Call Not Suck
- Don't follow the sun.
- Incident Analysis and Chaos Engineering: Complementary Practices
- Markers of Progress in Incident Analysis - How do you know you're actually learning from experience?

Remote Teams

- Awesome Remote Job - A big curated list of resources for remote jobs.
- Best Practices for Managing Remote Teams: A Psychological Perspective
- Managing Remote Teams - A Crash Course
- 12 Tips For Managing a Remote Team (And Loving It)
- Employee Engagement in a Remote World
- Better Remote Meetings - Tips to equalize participation if you're split between in-person and remote.
- Why remote work is inclusion work
- How to build social connection in a remote team
- Managing Remote Developer Teams: How Buffer Set the Gold Standard - Interview with Katie Womersley
- Remote Mob Programming
- Working with distributed teams - "To be able to work well with distributed teams, we need to pay very close attention to our communication, in order to always strive to convey our message as clearly as possible, and in doing so, avoid having different views regarding the same subject. In addition to effective communication, we also need to efficiently use the tools that are available for this work context."
- What is it like to work remotely as a software developer?
- Working from home: the yin and yang - Advice on dealing with the physical and mental challenges of being out of the office full-time.
- Tools That Make Work Faster - "...when developing software in a remote team"
- Asynchronous Communication: The Real Reason Remote Workers Are More Productive
- How to be a better remote worker
- The Manager’s Schedule Is Holding Back Remote Work
- What Most Remote Companies Don’t Tell You About Remote Work - "Isolation, anxiety, and depression in the remote workplace and what we’re doing about it"
- Managing From Afar: How This Engineering Manager Tackles the Challenges of Remote Work
- Long-Distance Relationship: How To Work With Clients You've Never Met Face-to-Face - Ebook from Toggl.
- Remote Work Encyclopedia
- Scaling Culture: Retain Your Developer Team’s Soul During Fast Growth
- A Distributed Meeting Primer
- A guide to distributed teams
- Working as a Remote Product Designer from far far away

- Are You Ready to Work Remotely? - Guidance for newer developers.
- How to Work From Home and Actually Get Stuff Done
- There's One Major Reason Remote Work Can Go Horribly Wrong
- The tools and tricks that let Ars Technica function without a physical office
- How I Work From Anywhere in the World - With advice on travel as well as hardware/software issues.
- Tracking the future of remote workplaces: Apps, communication, and liability - Crystal-ball gazing from Ars Technica
- Setting Up Your Webcam, Lights, and Audio for Remote Work, Podcasting, Videos, and Streaming
- Surviving meetings while remote
- Busting the common misconceptions about working from home
- The science behind self-care and practising it when working remotely
- Reworkin - Social networking site for remote workers.
- Open Offices Inhibit Remote Work
- Dear Founder: So You're Thinking About Building a Remote Team
- Working from home – things no one talks about
- Remote Work Can (And Does) Boost Employee Productivity
- How We Work - Report on five years of being fully remote from fournova.
- Working Remotely is a Skill
- The Coming Mental Health Crisis as Remote Working Surges
- Zippia Poll: Half of American workers would rather work from home forever

Staying Technical

- From Engineer to Manager: keeping your technical skills
- How (and why) Should Managers Code?
- The Fallacy of needing a technical manager

Team-Building and Scaling

- The Essential Guide to Building Balanced Development Teams - with a particular focus on junior/senior developer balance & mentorship
- Building Distributed Engineering Teams - Patterns and antipatterns from Doximity.
- Scaling Engineering Organizations - Stripe is way bigger than your organization, but there are some good thoughts on hiring, onboarding, and related topicshere.
- 10 Tips to Boost Developer Productivity in Your Team
- How to Create a Feedback Culture
- Introducing Engineering Management to a Growing Organization
- Designing Engineer Onboarding at Affinity
- Camille Fournier on Scaling, Structure, and Growing as an Engineering Manager\ - It's even more complex when the company is growing while you're getting more responsibility. Also has some material on composing an engineering track.

- How to evolve an engineering organization. - Advice on scaling up from someone who's been there.
- Scaling Technology and Organizations Together with Randy Shoup
- An Alternative Approach to Re-Orgs At Your Company
- Accelerate[How to build a generative engineering culture] - If you read (agile_scrum_xp_and_lean) and the idea of a Westrum generative culture resonated with you, here are some ideas of things you as a manager can do to encourage one to flourish.
- How to run a successful team retreat
- Organizing team offsites.

Values

- Engineering Values - A list of values for one team at Medium
- Just Culture - A healthcare quality concept that more software organizations should be familiar with.
- How we align our goals

Your Influence

- How to Make Sense of Your Impact When You're No Longer Coding
- Influence - Pretty much everything on the Engineering Manager blog is worth reading, but this one stood out
- Why Great Managers Matter
- Reasoning about Leverage in Engineering Organisations
- How to stop micromanaging and give your team autonomy
- Developer Burnout: How Can a Manager Spot It and Stop It
- It's Not All About You: Truly Empowering Your People

Beyond First-Line Manager

- The Evolution of Management - "Transitioning up the ladder"
- What do executives do, anyway? - Looking further up the ladder

Unsorted Links

- The Change Gap: How to Unleash Your Desired State
- How Google sets goals: OKRs
- How do managers get stuck? - Managing down, up, and sideways for greater impact & visibility
- The Geek's Guide to Leadership (video specifically aimed at team leads)
- Design Patterns for Managing Up - "Four challenging work situations and how to handle them"
- Learning by fixing — the value of triage engineer rotations
- Engineering Management Philosophies and Why They Matter Even if You are Not a Manager

- 11 Practical Steps Towards Healthy Power Dynamics at Work
- Fire Fixation: "Fire fixation is the pattern of patching urgent problems as quickly as possible, and then immediately moving on to the next."
- Targets and Estimations in Software Development - You need to understand the difference.
- Supermanagers
- “10x engineers”: Stereotypes and research
- Clearbit's approach to management
- Lessons from Stripe: What one manager learned about ambition, optimism, and recruiting.
- Starting an Engineering Management Book Club
- Lessons on leadership: The 10 most impactful lessons I’ve learned from 1,000+ managers in 2019
- How to build your company's engineering brand.
- Lessons learned managing the GitLab Data team - Reflecting on a year of management

Team Lead & Developer Resources

Many of these are aimed specifically at team leads, but there are more general articles as well. If you're a lead, you will probably also want to have a look at the Engineering Manager resources, particularly if you're leading a large team or multiple teams.

- Getting Toasty: Observations on Burnout - **If you're overwhelmed, depressed, or burned out, ask for help. Everyone else has been there too.**
- Global IT Burnout Index - Another good set of resources and a quick check of your own burnout level.

Code reviews

- A Guide to Mindful Communication in Code Reviews
- 7 Code Review Best Practices and Dynamics You Can Identify and Act On: Part 1
- 7 Code Review Best Practices and Dynamics You Can Identify and Act On: Part 2
- 10 Lessons Learned Conducting Code Reviews
- When Developers Disagree
- Code Review—The Ultimate Guide
- How to do a code review - A section of Google's engineering practices documentation.
- Keep Code Review from Wasting Everyone's Time
- How to Make Good Code Reviews Better
- The Engineering Manager's Guide to the Code Review Process - From GitPrime
- How to do High-Bar Code Review Without Being a Jerk

Culture

- How to Create a Great Team Culture (and Why It Matters)
- the Origins of Opera and the Future of Programming - "You don't hire star developers, put them together, and poof get a great team. It's the other way around. When developers form a great team, the team makes us into great developers."
- Do Software Developers Normally Code on Weekends? Work-life Balance and Overtime in the Tech Industry

Documentation, Communications and Status

- Want to be happier and more successful? Write about yourself every week.

- Making Engineering Team Communication Clearer, Faster, Better - Patterns for design documents & design reviews "Most engineering teams rely on design documents to describe, scope, and approve projects or features. Those aren't new in and of themselves (they're just the foundation of what we'll talk about here). Kicking off a project without one would be like a hiker heading into the forest without a map. Engineering teams don't have that kind of time."
- What nobody tells you about documentation - "There is a secret that needs to be understood in order to write good software documentation: there isn't one thing called documentation, there are four. They are: tutorials, how-to guides, explanation and technical reference. They represent four different purposes or functions, and require four different approaches to their creation. Understanding the implications of this will help improve most software documentation - often immensely." Blog post from Divio.

Ethics

- It's time programmers talked about ethics

Mentoring

- On Helping Junior Developers - "Communication Styles and Why Juniors Should Pick Their Own Mentors"
- Micro-promotions and mentorship: the big impact of small actions in an engineering culture
- [[https://blog.pragmaticengineer.com/developers-mentoring-other-developers/| Developers mentoring other developers: practices I've seen work well]]
- Heroes and Juniors: Increasing Engineering Team Velocity
- Ten Principles for Growth as an Engineer

OKRs

- OKRs for your Engineering Team
- Hate OKRs? Avoid these 7 mistakes
- Google's OKR Playbook
- OKRs: Potential Issues and How to Deal with Them
- Guide:Set Goals with OKRs
- The Beginner's Guide to OKR
- The Pros and Cons of Individual OKRs
- OKRs Aren't Going to Fix Your Communication Issues
- OKRs from a development team's perspective
- Alignment through OKR's and Hypotheses

Pair Programming

- Tuple's Pair Programming Guide
- On Pair Programming - Comprehensive roundup from ThoughtWorks.

Planning and Stories

- Agile Story Essentials - A visual aid going through one way of thinking about user stories.
- Everyone Should Read Customer Support Emails
- Do This Now: 8 Ways to Focus your Product Team on Impact, Not Features
- How to Write Good User Stories
- Coaching Tools – The Narrative
- Products Over Projects - "Software projects are a popular way of funding and organizing software development. The organize work into temporary, build-only teams and are funded with specific benefits projected in a business case. Product-mode instead uses durable, ideate-build-run teams working on a persistent business issue. Product-mode allows teams to reorient quickly, reduces their end-to-end cycle time, and allows validation of actual benefits by using short-cycle iterations while maintaining the architectural integrity of their software to preserve their long-term effectiveness."
- Definition of Done Examples for Software Projects - "Definition of Done as a shared understanding of what it means for work to be complete. To be honest, each agile team has its own Definition of Done. Basically, a team agrees on, and displays somewhere in the team room or in slack, google drive, whatever, a list of criteria which must be met before a product increment, normally it is a user story and there should be a clear definition of when it is considered "done"."
- What Happens & When During a Sprint - "Successful Scrum implementations involve a handful of important ceremonies. This includes sprint planning, the sprint review, the daily scrum, the sprint retrospective and more. There's often a lot of confusion about who participates, when these ceremonies are conducted, how long each can take, the purpose of the ceremony, and more. To reduce the confusion, we've created infographics that answer each of these questions for sprints of 1-, 2-, 3- and 4-weeks."
- Startups: How to Use Amazon's Narrative Process to Set Goals and Think Clearly
- Yes, You Should Estimate Software Projects
- Why Standups are Useless and How to Run Great Product Team Meetings - I don't think standups are useless, but it's worth thinking about why and how you do them.
- Makers, Don't Let Yourself Be Forced Into the 'Manager Schedule'
- What's the Difference between a Project and a Product?
- Results vs. Hours: creating a results-focused workplace
- Managing product requests from customer-facing teams: top 2 things - "A simple technique to maintain a reasonable backlog while making customer-facing teams feel empowered"
- 15 Questions to Ask at the Start of a New Software Project

Self-organizing Teams

- Designing Autonomous Teams and Services - "Modern, high-performing organizations employ continuous discovery and delivery to develop better products faster than their competitors. They are constantly running experiments to discover innovative new ways to solve customer problems, and they build high-speed engineering capabilities to deliver value every day, creating ultra-short customer feedback cycles."

Technical Debt

- Technical Debt Is Like Tetris
- What Ticketmaster is doing about technical debt - "This post describes the journey Ticketmaster has been on over the last year to define and measure technical debt in its software platforms." Patterns for mapping, reporting on, and tracking remediation of technical debt
- The Technical Debt Myth - Conflating many issues under this one heading makes them all harder to tackle. Are you *really* facing technical debt?
- How to tackle technical debt
- How to invest in technical infrastructure.
- The ONE chart every developer MUST understand - "This chart says that most teams could deliver their software projects sooner if they focused more effort on defect prevention, early defect removal, and other quality issues."
- How to Make Things High-Quality - If it's not worth doing well, it's not worth doing at all.
- Technical Debt is Soul-crushing
- Tech Debt and the Pragmatic Middle Ground

Technical Leadership

- Taming the Rate of Change - Developing a culture of safety so you can simultaneously get frequent deploys and high quality.
- All the best engineering advice I stole from non-technical people
- Papers We Love - "Papers We Love is a collection of academic computer science papers and a community who loves reading them."
- An incomplete list of classic papers every Software Architect should read - Links to lots of good stuff starting with Turing's “On computable numbers with an application to the Entscheidungsproblem”
- The Product-Minded Software Engineer - "At companies building world-class products, product-minded engineers take teams to a new level of impact."
- A Senior Engineer's CheckList
- 15 Easy Questions for Easy Change Management - "If your systems keep breaking when changes are made, answer these questions." A good checklist for major deployments.
- Thriving on the Technical Leadership Path
- The Leadership Library for Engineers - Huge list of resources
-

    - Technical Lead Management 101 or How to Try Out Management

Testing

- Testing in Production: the hard parts

When Things Go Wrong

- The Pre-Mortem: A Simple Technique To Save Any Project From Failure
- It's about what broke, not who broke it - "The important thing at the time was that we cared about what broke, not who broke it. Who broke it is frequently just a roll of the dice: who got that particular task, bug, or ticket assigned to them, and happened to run this valid command instead of that also-valid command? Why would you ever assign blame based on that?"
- https://medium.com/tock/why-blameless-post-mortems-80f9f446fb77

Unsorted Links

- The Conjoined Triangles of Senior-Level Development - "The simplest explanation of seniority across companies is this: *How much direction will this person need, and how much will they be able to provide to others?*"
- On Being A Principal Engineer - "I also realized that while I am still an individual contributor, the principal engineer role carries enough cross-organization work, and enough people skills, that it is much closer to management than it may seem without engineers reporting directly to me."
- Cognitive Biases in Programming - As developers, we’re familiar with the various problems that interfere with our productivity. But often we overlook the broad picture. Some subtle, some huge, some you can do something about, and some you just, well, can’t. These all combine to form a sort of internal feedback loop that can lead to lost hours of productivity, bugs, and just all-around frustration. If we can minimize the impact of one or two of these, we can break the cycle and neutralize the rest.
- The Effective Tech Lead is a 100x Engineer - The Webflow Tech Lead Guide
- Techie to tech lead: My five biggest mistakes - Some advice from a current Head of Technology at ThoughtWorks: "As a young, ambitious developer with a strong sense of my own talent, I was eager to become a tech lead, and it took less than four years for me to achieve this goal. But over the next two years, the experience and reality of leading a team put me off leadership completely. For several years after, I retreated into the security of the technology, shunning any opportunity to take on more responsibility. Over time, I gained an understanding of the root causes of my mistakes and eventually regained the confidence to accept new leadership opportunities and grow as a leader, and with the right support over the last two years, I've grown as a Head of Technology inside ThoughtWorks. As I have coached and mentored other leads, I’ve learned that some of my mistakes weren't

unique to me but are common among technologists who move into leadership."

- How to Grow as an Engineer (Working Remotely) - "I have over 20 years of experience as an engineer or engineering manager. I've spent the last seven years working for The New York Times, and have been remote for the past three years. How do I keep myself constantly learning and growing, both personally and professionally, while at the same employer, especially now that I'm a remote worker?"
- Being Glue - How to handle it if you move into a less-coding but still-technical "connector" role
- DACI Playbook]]from Atlassian
- Software Development is more like a professional competition than building a house]] - Metaphorical exploration from Christopher Drappier

Engineering Hiring

How to Hire

- Buiding Intersectionality Into Your Hiring Strategy
- Great developers are raised, not hired - Trying to hire "rock stars" is a waste of time.
- How to make your organization attractive to engineering talent
- How NOT to hire a software engineer
- You probably don’t factor in engineering time when calculating cost per hire. Here’s why you really should.
- How Splice Builds Globally Distributed Engineering Teams
- How to Fix Your Tech Interview to Increase Diversity
- Assholes: A Probing Examination - "Whatever you do, don’t hire assholes at your company."
- Want to hire the best programmers? Offer growth.
- The Ultimate Guide To Startup Hiring - Hiring more than just engineers
- Hiring is Broken and Yours is Too - Pros and cons of all the common techniques for hiring developers.
- I never liked technical interviews
- How we doubled the representation of women in Engineering at Clio
- Eric's Guide to Hiring {Software Developers}
- What's Wrong with the Tech Interview Process? - A look at some of the issues, along with questions to ask yourself if you're trying to hire.
- The Qualified Manifesto On Hiring Software Developers
- Reducing the Impact of Unconcious Bias in Our Hiring Process
- How to Make Tech Interviews a Little Less Awful - "What is your goal in interviewing candidates? If you answered 'to hire the best candidate' you should reconsider. Forming the best team is a goal that will better serve your company."
- Where 75% of workers are on the autistic spectrum
- Evolving our Interview Process
- Improving How We Interview
- We only hire the trendiest - A stupid (but common) way to limit your candidate pool.
- No engineer has ever sued a company because of constructive post-interview feedback. So why don’t employers do it?
- The Horrifically Dystopian World of Software Engineering Interviews
- The Experience Paradox
- The Science Behind Making Software Engineering Interviews Truly Predictive of Job Performance
- Thoughts on Recruiting
- Take-home vs whiteboard coding: The problem is bad interviews
- The software industry's greatest sin: hiring

Take-Homes and Challenges

- 3 good reasons to kill whiteboard and design challenges in interviews
- Design challenges in interviews — how not to?
- Why We Prefer Coding Challenges and How We Made Ours Better - If you're going to require candidates to write serious code, you owe it to them to make the process as good as possible.

How to Get Hired

- Don't help me say No - Advice to applicants to avoid common mistakes.
- Advice to Less-Experienced Developers - From Mike Gunderloy
- Interviewing Red Flags - From Mike Gunderloy
- Interviewing at Senior Levels - Advice from Lara Hogan
- Reverse Interview - Questions you should ask the company
- What to ask during your interview - Another selection of reverse interview questions.
- Job negotiation for programmers: the basic principles
- Tech Hiring Managers Answer Common Behavioral Interview Questions
- Questions to ask at the end of a technical interview
- Questions To Ask During An Interview As a Developer

Books

- The Effective Hiring Manager - I grabbed this one as soon as it was available and I'm glad I did; a lot of Horstman's advice will inform my future hiring process. That said, many many people are going to hate this book because it is bluntly about doing a good job of hiring based on job fit and merit. If you're one of the people who feels diversity & inclusion must be considered when hiring, well, those words aren't even in the index. There are also other times that you'll want to bring someone in based on factors other than proven ability in job skills (hiring interns, for example). But if you want a step-by-step process that leads to efficient decision-making and removes dependence on "gut feeling", and that goes from reading CVs straight through onboarding, you've come to the right place. (Reviewed by Mike Gunderloy)

Architecture

Architecture

- Questions for a new technology - A framework for deciding when to bring something new into the stack. **Read this before following the cool kids down the latest rabbit hole.**
- The Agile Manifesto: A Software Architect's Perspective
- Distributed systems vocabulary.
- https://herbertograca.com/2019/08/12/documenting-software-architecture/
- Describing fault domains.
- The Architecture of Open Source Applications - A bunch of case studies
- Software Architecture Guide - Martin Fowler has published much excellent material on architecture. This page provides an overview and entry point.
- Software Architecture is Overrated, Clear and Simple Design is Underrated - I think this is more of an argument against "architecture theater" than appropriate architecture.
- What is a Software Architect? - One architect's view & study guide.
- The Myth of Architect as Chess Master - What architects are not.

Front-end/UI

- Prisma - Generalized GraphQL/REST middleware layer in JS. Potentially useful for combining responses from multiple microservices.
- How to make your web app work offline - "Service workers, caches, IndexedDB and PWA."

Back-end

- On Sharding
- Eventing Facets - Excellent series on eventing and messaging services from Tim Bray.

Operations

- Evolving Regional Evacuation - How Netflx manages resilience at scale.
- Observations on Observability

AWS

- The Open Guide to Amazon Web Services - Tons and tons of information. Start here.

- [[https://www.trek10.com/blog/Trek10 Blog]] - Blog from an AWS-focused consulting firm

Security

- Zero trust architecture design principles - I feel like this chunk of architecture is becoming increasingly important.

Databases

PostgreSQL

- postgresqltuner - ""postgresqltuner.pl" is a simple script to analyse your PostgreSQL database. It is inspired by "mysqltuner.pl"."
- Vertically Scaling PostgreSQL
- Active Record Extended - "Active Record Extended is essentially providing users with the other half of Postgreses querying abilities. Due to Rails/ActiveRecord/Arel being designed to be DB agnostic, there are a lot of left out features."

MySQL/MariaDB

- innotop - "top"-like performance monitoring specifically for InnoDB

ML/AI

- Python Data Science Handbook - "This repository contains the entire Python Data Science Handbook, in the form of (free!) Jupyter notebooks."

Serverless & AWS Lambda

Articles & Blog Posts

General Serverless

- Five Facets of Flow Strategy - A broad set of things to consider as we move into a serverless, event-driven network of small pieces.
- Serverless Best Practices
- Serverless Failure Stories
- Serverless Architectures
- Serverless Mullet Architectures
- Getting Started with the PLONK Stack and Serverless 2.0 - Prometheus, Linkerd, OpenFaaS, NATS, Kubernetes

AWS Lambda

- Serverlessness- Series of blog posts from Amazon's Tim Bray, based on his re:invent 2018 talk.
- Lambda optimization tip – enable HTTP keep-alive - For DynamoDB specifically
- AWS: How to limit Lambda and API Gateway scalability: Cost control in the cloud.
- Best Practices for AWS Lambda Container Reuse: "Optimizing Warm Starts When Connecting AWS Lambda to Other Services"
- A useful tool for building serverless Ruby apps with AWS Lambda: Introducing Ruby_Lambda.
- https://aws.amazon.com/blogs/architecture/ten-things-serverless-architects-should-know/
- Lambda optimization tip — enable HTTP keep-alive - Including notes on how to test the optimization
- Designing durable serverless apps with DLQs for Amazon SNS, Amazon SQS, AWS Lambda
- AWS Serverless WebSockets — Introduction Around the Pitfalls
- AWS Lambda the CLI Way - In-depth tutorial that does everything from the command line.

OpenFaaS

- Meet faasd - portable Serverless without the complexity of Kubernetes

Development

- Serverless Functions With WebAssembly Modules
- FaaS for the Rubyist - Self-hosting with OpenFAAS.
- Serverless-Dev-Tools - Console-based management for AWS Lambda.

- 24 open source tools for the serverless developer: Part 1 and Part 2

Security

- flAWS2.cloud -Interactive security training that lets you play the attacker or defender roles.

Logging

- Introducing Datadog for serverless

AWS API Gateway

- A Detailed Overview of AWS API Gateway

Deployment

- How to use CloudFormation to deploy Frontend Apps to S3 and Serverless Application Repository: A nice workaround for some AWS limitations.
- Serverless AppSync Component: "The AppSync Serverless Component allows you to easily and quickly deploy GraphQL APIs on AWS, and integrate them with AWS Lambda, DynamoDB & others."

Testing

- Fighting vendor lock-in and designing testable serverless apps using hexagonal architecture

Other Tools

- Lambda Warmer - Optimizer for Lambda cold starts
- Announcing Ruby build support for AWS SAM CLI

Design (UI and otherwise)

Accessibility & Related Topics

- Inclusive Design - Good resources from Microsoft.
- Ethical Design

Microservices

Some links that might be useful as we move forward into a microservicesworld:

Sites

- Microservices Resource Guide - from Martin Fowler
- Microservice Architecture - Big collection of patterns and sample code, from the author of *Microservice Patterns* (see the Books section below)

Articles & Blog Posts

General

- Notes on Distributed Systems for Young Bloods
- Christian Posta - Architect at Red Hat who writes quite a bit about microservice issues
- Zipkin - Distributed tracing system
- Microservices are hard—an invaluable guide to microservices.
- Interview with Scott Bellware (Eventide co-creator) about micro-services in Ruby, event sourcing, autonomous services, SOA, dumb pipes, EventStore and mis-representation of terms in IT
- The Death of Microservice Madness in 2018 - This one has guidelines for deciding whether you're ready for microservices
- Microservice Madness - Quick (and shallow) look at some microservice problems
- Containers Will Not Fix Your Broken Culture (and Other Hard Truths)
- Microservices in Adopt? - Perspective from ThoughtWorks about what it takes to succeed with microservices
- Microservices in a Post-Kubernetes Era
- Micro-Services, and Co-ordination
- Why should you use microservices and containers? - Pros and cons according to IBM
- On Infrastructure at Scale: A Cascading Failure of Distributed Systems - A cautionary tale with some general advice
- The Cost of Bad Software Architecture
- Microservices After Two Years
- Services are not a Silver Bullet
- Monolith vs. Microservices
- Microservices, Containers and Kubernetes in 10 minutes
- Microservices Done Right, Part 2: More Antipatterns to Avoid
- Microservices Done Right, Part 3: Quantifying the Benefits of Microservices
- The Fargate Illusion
- 9 tips for moving code to microservices - Big project advice from HPE.
- Container Misconceptions - Some common ones

- Why Microservices Should Scare You More
- Forget monoliths vs. microservices. Cognitive load is what matters.
- Guiding Principles for Developer Tools - When you're big enough to be building internal services for other groups of developers.
- Are you sure you're using microservices? - Some warning signs that you may be building a distributed mess instead of microservices.
- So you want to learn Microservices? - Another big collection of links.
- What are the essential skills for Microservices developers? - Tech and soft skills

Keeping a Monolith

- Goodbye Microservices: From 100s of problem children to 1 superstar - Segment backed away from microservices.
- To Microservice or Monolith, that is the question... - "unless you have development resources spilling over the gunnels, stick with a monolith."
- Well Architected Monoliths are OK
- Microservices Are Something You Grow Into, Not Begin With
- The Neomonolith - Building a monolith with the intent of making it microservices later
- Scaling the Monolith - Maybe we don't need microservices after all.
- Deconstructing the Monolith: Designing Software that Maximizes Developer Productivity - Shopify uses the "modular monolith" pattern
- MicroServices - Check Size Before Ordering
- Give Me Back My Monolith
- Istio as an Example of When Not to Do Microservices
- Monoliths are the future
- The dark side of microservices

Decomposing into Microservices

- Break that big ball of mud!
- The great microservices migration - Slide deck based on Uber's experience
- Should that be a Microservice? Keep These Six Factors in Mind - Some guidance from Pivotal
- How do you cut a monolith in half?
- Do you have too many microservices? - Five Design Attributes that can Help - Another look at how to split up a monolith
- You're Not Actually Building Microservices - Thoughts on avoiding the distributed monolith problem
- Strategies for Decomposing a System into Microservices - Some heuristics for finding the boundaries
- The misguiding mindset that you are using and how it is hindering your microservices architecture
- Monoliths to Microservices - A cartoon view from Pivotal
- Is a Shared Database in Microservices Actually an Anti-pattern?
- YAMVM—Yet Another Monolith vs. Microservices
- Mono's Journey from monolith to microservices
- 3 Strategies for implementing a microservices architecture

- Using Microservices to Solve Slow Build Times - A strategy for getting started.
- Faster, cheaper, and better: A story of breaking a monolith - A detailed case study
- Microservice Architecture at Medium
- How to sleep at night having a cloud service: common Architecture Do's
- Should this be a microservice?
- Data-Oriented Architecture - "In data-oriented architecture, a monolithic data store is the sole source of state in the system, which is being acted on by loosely-coupled, stateless microservices."
- Untangling Microservices, or Balancing Complexity in Distributed Systems - Avoiding the distributed big ball of mud

Containerization

- A Practical Introduction To Container Terminology - From Red Hat

Frameworks

- Eventide - A full framework for event-sourced autonomous services in Ruby
- Rails Event Store - "The open-source implementation of an Event Store for Ruby and Rails"
- Announcing Distributed Application Runtime (Dapr), an open source project to make it easier for every developer to build microservice applications - Framework from Microsoft
- Go Micro - "Go Micro is a framework for microservice development." Written in Go, of course.

Event Sourcing

- Event Sourcing Made Simple
- What They Don't Tell You About Event Sourcing
- An In-depth Look at Event Sourcing with CQRS - 461-slide presentation
- Datomic: Event Sourcing without the hassle
- Don't Let the Internet Dupe You, Event Sourcing is Hard
- CQRS and Event Sourcing Intro For Developers
- Mistakes we made adopting event sourcing (and how we recovered)
- 1 Year of Event Sourcing and CQRS
- Event Sourcing - Good overview

APIs & Gateways

- Design Patterns in API Gateways and Microservices - With a list of cross-cutting concerns
- Rate Limiting for API gateways
- Gubernator: Cloud-native distributed rate limiting for microservices - An open-source tool from Mailgun.
- API Gateways are going through an identity crisis
- Microservice API Patterns

- Everything You Need To Know About API Rate Limiting
- Architecting multiple microservices behind a single domain with Amazon API Gateway
- ExcessFlow - "ExcessFlow is a high precision Redis based rate limiter; it means that even with hundreds and even thousands requests coming in all at once it will not allow an occasional request slip over limit (causing potential race conditions or unwanted extra invocations of your code)."
- How to communicate your Microservices? - Choices for external APIs.

Messaging

- Microservices Messaging: Why REST Isn't Always the Best Choice
- Messages on Rails Part 1 - Introduction to Kafka and RabbitMQ
- Messages on Rails Part 2: Kafka
- Messages on Rails Part 3: RabbitMQ
- 7 Commandments for Event-Driven Architecture
- Why We're Switching to gRPC
- Simple Two-way Messaging using the Amazon SQS Temporary Queue Client
- Introduction to Event-driven Architectures With RabbitMQ
- Announcing Message DB: Event Store and Message Store for PostgreSQL

Networking

- We built network isolation for 1,500 services to make Monzo more secure - A tale of rearchitecting towards a zero-trust network.

Service Mesh

- How did we get to service meshes?
- Envoy Proxy at Reddit
- Comparing Kubernetes Service Mesh Tools
- Containers, microservices, and service meshes: Comparisons & lessons extracted from a real-world project.
- Announcing Maesh, a Lightweight and Simpler Service Mesh Made by the Traefik Team
- The Service Mesh: What Every Software Engineer Needs to Know about the World's Most Over-Hyped Technology - From one of the original Linkerd developers.

Patterns

- Future-Proofing Backend Services with GraphQL, PostgreSQL and Docker (Part 1)
- Every service is an island - Design pattern to help prevent cascading failures
- Embracing context propagation
- The Circuit Breaker Pattern
- The architecture of declarative configuration management
- Application integration patterns for microservices: Fan-out strategies - AWS-specific architectural advice

Testing

- Testing Microservices, the sane way - One of the best online articles I've ever read
- Testing of Microservices - Spotify engineers argue for testing edge behavior in isolation
- Integrated Tests Are A Scam - Don't write tests that pass or fail depending on spinning up other services
- Five microservice testing strategies for startups
- The Cloud Native QA
- Break functional and orchestration responsibilities for better testability
- Tinkertoys, Microservices, and Feature Management: How to build for the future - Slides from @wiredferret
- micro-jaymock - "Tiny API mocking microservice for generating fake JSON data."

Contract-based testing

- Explicit Contracts for Rails
- Pacto
- pact-ruby

Performance

- Microservices and Availability
- Reasons to Scale Horizontally
- The Patterns of Scalable, Reliable, and Performant Large-Scale Systems

Security

- Securing Microservices (Part I)
- Learn how to use JSON Web Tokens (JWT) for Authentication
- Micro-services Architecture with Oauth2 and JWT – Part 1 – Overview
- Why JWTs Suck as Session Tokens
- API Security Checklist: "Checklist of the most important security countermeasures when designing, testing, and releasing your API."
- How to think about Zero Trust architectures on AWS
- JWT is Awesome: Here's Why

Deployment

- A production-grade CI/CD Pipeline for Serverless Applications - Specifically using GitLab and AWS Lambda
- How to Design Services for Continuous Deployment: 3 Best Practices
- In the Loop: How a Release Team Centralizes and Aligns Processes
- Docker Deployments using Terraform
- Continuous Delivery Sounds Great, but Will It Work Here?
- Measuring and Improving your CI/CD Pipelines
- Canary analysis: Lessons learned and best practices from Google and Waze

- Maybe You Don't Need Kubernetes - Using Nomad for orchestration.

Front-End

- Micro Frontends
- Front-end Microservices at HelloFresh
- Front-End Micro Services - Zalando's approach to composing HTML fragments while maintaining consistency
- Micro Frontends
- Microfronts - "Polyglot Front-End Solution for running multiple frameworks as one"
- Micro Frontends Architecture
- Microfrontends For The Win!

Testing Tools

- Insomnia
- Paw
- Pumba - Chaos testing for Docker containers

Other Tools

- Skipper - Router & reverse proxy for microservice composition
- Mockoon - "Mockoon is a free and open source local server/API mocking tool built with Electron"

Conference Talks

- Migrating to Microservices Databases - patterns for zero-downtime database schema changes, back & forward compatibility, how to split the database
- SLO BURN - A Google SRE discusses SLOs and error budgets and monitoring and related topics.

Books

- The Tao of Microservices - Excellent.
- Migrating to Microservice Databases: From Relational Monolith to Distributed Data - Free, requires registration with Red Hat Developer Program. Good discussions of zero-downtime migration patterns and ways to integrate data across multiple services. Strongly recommends one database per microservice.
- Microservices for Startups - Free, requires email registration with ButterCMS to get chapters emailed to your inbox. Or you can wait for them to show up on the web page; first two chapters are available as of January 8.
- Microservices Antipatterns and Pitfalls - Free, requires email registration with O'Reilly to download

- Microservice Patterns - Currently in the Manning Early Access Program, free download for chapter 1
- Building Microservices: Designing Fine-Grained Systems
- Production-Ready Microservices
- Designing Distributed Systems - O'Reilly ebook, free copy from Microsoft with email registration
- Practical Microservices: Build Event-Driven Architectures with Event Sourcing and CQRS

Training

- An introduction to distributed systems - Course notes from a class that covers a lot of ground

Humor

- Microservices - Three-minute video sketch from KRAZAM.

Software Development Life Cycle

- Why deadlines and sprints are bad for you
- Agile as a Corollary to Twyman's Law - "If most upfront requirements are non-value adding, then learning your way forward is more important than executing a plan. That, by the way, is Agile in a nutshell."
- Understanding Fake Agile

Security

Threat Modeling

- Threat Modeling: 12 Available Methods
- Attack matrix for Kubernetes - A worked example of using the MITRE ATT&CK framework

Monitoring

- Putting AWS Security Services to Work for You