CAMILLE FOURNIER

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Camille Fournier - 双语对照

This is the complete bilingual transcript of Lenny's Podcast featuring Camille Fournier.

(00:00:00) Lenny Rachitsky

English:

I'm curious what it is that PMs do that annoy engineers most.

中文翻译:

我很想知道，产品经理（PM）做的哪些事情最让工程师感到厌烦。

(00:00:04) Camille Fournier

English:

Hoarding credit. PMs, they tend to be the front-facing person for initiative. Engineers sometimes think that they don't get the credit for their work because the PM takes all the glory and all the credit for the project that they really worked very hard on.

中文翻译:

独揽功劳。PM 通常是项目的对外窗口。工程师有时会觉得自己的工作没有得到应有的认可，因为 PM 拿走了所有的荣誉和功劳，而这些项目其实是工程师们付出了巨大努力才完成的。

(00:00:19) Lenny Rachitsky

English:

I find the best PMs are the ones that talk the least and encourage other people to do the presenting-

中文翻译:

我发现最优秀的 PM 往往是那些话最少，并鼓励其他人去进行演示和汇报的人——

(00:00:23) Camille Fournier

English:

The next thing that engineers really get annoyed about with PMs, when they just don't understand the details and act like they don't matter, it just shows a real lack of empathy for the work that engineers are doing and I think it really can be very off-putting.

中文翻译:

下一件让工程师对 PM 感到非常恼火的事，就是 PM 完全不了解细节，还表现得好像细节根本不重要。这表现出对工程师所做工作的极度缺乏同理心，我认为这真的非常令人反感。

(00:00:34) Lenny Rachitsky

English:

Is there any insight you can give about what people may be missed about the motivation of engineers, what gets them excited?

中文翻译:

关于人们可能忽视的工程师动力来源，或者说是什么让他们感到兴奋，你有什么见解吗?

(00:00:40) Camille Fournier

English:

A lot of people assume that engineers just write code and don't underestimate the ability for your engineers to want to understand the business problem, want to understand the customer problem. I think the product managers that have done the best, they're not threatened by other people having ideas.

中文翻译:

很多人认为工程师只是写代码的，但千万不要低估工程师想要理解业务问题和客户问题的愿望。我认为做得最好的产品经理，是那些不会因为别人有想法而感到受威胁的人。

(00:01:00) Lenny Rachitsky

English:

Today, my guest is Camille Fournier. Camille is one of the most respected technology executives in tech and the author of the Manager's Path, which many considered the definitive guide for navigating your career and moving into management. Over the course of her career, she was CTO of Rent The Runway, VP of technology at Goldman Sachs, global head of engineering and architecture at JP Morgan Chase and head of platform engineering at Two Sigma. She's also releasing a new book later this year called Platform Engineering, A Guide for Technical Product and People Leaders, which you can actually pre-order today and we get into this topic in the latter half of the conversation.

中文翻译:

今天的嘉宾是 Camille Fournier。Camille 是科技界最受尊敬的技术高管之一，也是《管理者之路》（The Manager's Path）一书的作者，许多人认为这本书是职业发展和转型管理的权威指南。在她的职业生涯中，她曾担任 Rent The Runway 的首席技术官（CTO）、高盛（Goldman Sachs）的技术副总裁、摩根大通（JP Morgan Chase）的全球工程与架构负责人，以及 Two Sigma 的平台工程负责人。她还将在今年晚些时候发布新书《平台工程：技术产品与团队领导者指南》（Platform Engineering, A Guide for Technical Product and People Leaders），现在已经可以预订了，我们会在对话的后半部分深入探讨这个话题。

(00:01:36) Lenny Rachitsky

English:

We also dig into what PMs do that most annoys engineers and how to stop doing these things. Why major rewrites are often a trap. Why you may want to be doing fewer one-on-ones. What most surprises people when they become a manager and some really useful heuristics for how long you should stay in IC before you make the leap into management and tons more. This episode covers a lot of ground and we'll help you think about management. platform teams, team culture and the PM and end relationship in a whole new way. If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. It's the best way to avoid missing future episodes and helps the podcast tremendously. With that, I bring you Camille Fournier. Camille, thank you so much for being here, welcome to the podcast.

中文翻译:

我们还将探讨 PM 做的哪些事最让工程师讨厌，以及如何停止这些行为。为什么大规模重写（rewrites）往往是一个陷阱。为什么你可能需要减少一对一面谈（1:1s）。人们在成为管理者时最感到意外的事情，以及关于在转型管理前应该做多久个人贡献者（IC）的一些非常实用的启发式建议，还有更多精彩内容。本集涵盖了广泛的领域，将帮助你以全新的方式思考管理、平台团队、团队文化以及 PM 与工程师的关系。如果你喜欢这个播客，请不要忘记在您常用的播客应用或 YouTube 上订阅和关注。这是避免错过未来节目的最佳方式，也对本播客有巨大帮助。现在，让我们欢迎 Camille Fournier。Camille，非常感谢你能来，欢迎来到本播客。

(00:02:23) Camille Fournier

English:

Thank you so much for having me.

中文翻译:

非常感谢邀请我。

(00:02:25) Lenny Rachitsky

English:

It's my pleasure. I want to start by asking you a question that is on the minds of a lot of product managers is how to be less annoying as a product manager. I know you worked with a lot of engineers over time. I'm curious what it is that PMs do that annoy engineers most and how can PMs stop doing that?

中文翻译:

这是我的荣幸。我想先问一个很多产品经理都在思考的问题：作为产品经理，如何才能不那么招人烦？我知道你长期以来与很多工程师合作过。我很想知道 PM 做的哪些事最让工程师讨厌，PM 又该如何改进？

(00:02:42) Camille Fournier

English:

I would say there are a few things that PMs do that annoy engineers and to be clear, I am sure that engineers annoy PMs, just as much. So I've realized this is a two-way street. So I think there's some things that are really easy to fix and some things that are maybe a little bit harder. So the easy things to fix are hoarding credit. Sometimes I think PMs because they tend to be the front-facing person for initiatives, they're talking to customers, they're talking to the executive team, whatever. Engineers sometimes think that they don't get the credit for their work because the PM takes all the glory and all the credit for the project that they really worked very hard on, right? So making every effort to be credit sharing and inclusive of the engineering team and giving them the opportunity to speak about their contributions

when it makes sense. I think those are all things that PMs can do to avoid that kind of ... What I consider a pretty easy annoyance, just like don't pour it all the credit. This is not just you, right? There's a lot of work has to go into that.

中文翻译:

我想说 PM 确实有几件事会让工程师抓狂，当然，我也确信工程师让 PM 抓狂的程度也不相上下。所以我意识到这是相互的。我认为有些事情很容易解决，有些则难一些。容易解决的是“独揽功劳”。有时我觉得 PM 因为是项目的对外负责人，他们要和客户谈，和高管团队谈等等。工程师有时会觉得自己的工作没得到认可，因为 PM 拿走了项目所有的光环和功劳，而工程师其实付出了极大的努力。所以，尽一切努力去分享功劳，包容工程团队，并在合适的时候给他们机会讲述自己的贡献。我认为这些都是 PM 可以做的，以避免这种……我认为相当容易解决的烦恼，就是别独吞功劳。这不只是你一个人的功劳，对吧？背后有很多工作。

(00:03:26) Camille Fournier (Continued)

English:

I think that sort of dovetails into the next thing that engineers really get annoyed about with PMs when they just don't understand the details and act like they don't matter and I think this is just a little bit of a cultural difference. I mean, even managers, just normal managers, people like me who are looking across really broad areas or you have to be kind of big picture focused, and you forget that engineering done successfully really is all about the details and you don't necessarily have to understand all of those details, but when you act like they don't matter and you don't care about them and it's just like, I don't care, just like tell me when you can get this done or why is it going to take so long? My god, this just seems like such a little thing. It just shows a real lack of empathy for the work that engineers are doing and I think it really can be very off-putting. Even though I will totally agree that sometimes you're going to get details that don't really matter and you just have to be a little bit patient in those circumstances.

中文翻译:

这紧接着引出了下一件让工程师非常恼火的事：PM 完全不理解细节，还表现得好像细节无关紧要。我认为这其实是一点文化差异。我的意思是，即使是管理者，普通的管理者，像我这样负责广泛领域或者必须专注于大局的人，有时也会忘记：工程的成功完全取决于细节。你不需要理解所有的细节，但当你表现得好像它们不重要、你不在乎，只是说“我不管，告诉我什么时候能做完”或者“为什么要花这么长时间？天哪，这看起来就是件小事”，这表现出对工程师工作缺乏真正的同理心，我认为这非常令人反感。尽管我也完全同意，有时你会听到一些确实不重要的细节，在这种情况下你只需要保持一点耐心。

(00:04:50) Lenny Rachitsky

English:

Today's episode is brought to you by DX. If you're an engineering leader or on a platform team, at some point your CEO will inevitably ask you for productivity metrics, but measuring engineering organizations is hard and we can all agree that simple metrics like the number of PRs or commits doesn't tell the full story. That's where DX comes in. DX is an engineering intelligence solution designed by leading researchers, including those behind the DORA and SPACE frameworks. It combines quantitative data from developer tools with qualitative feedback from developers to give you a complete view of engineering productivity and the factors affecting it. Learn why some of the world's most iconic companies like Etsy, Dropbox, Twilio, Vercel and Webflow rely on DX. Visit DX's website at getdx.com/lenny.

中文翻译:

本集节目由 DX 赞助。如果你是工程领导者或平台团队成员，CEO 迟早会向你索要生产力指标。但衡量工程组织很难，我们都同意像 PR（拉取请求）数量或提交数（commits）这样的简单指标无法说明全貌。这就是 DX 的用武之地。DX 是由领先的研究人员（包括 DORA 和 SPACE 框架背后的专家）设计的工程智能解决方案。它将来自开发工具的定量数据与来自开发者的定性反馈相结合，为你提供工程生产力及其影响因素的完整视图。了解为什么像 Etsy、Dropbox、Twilio、Vercel 和 Webflow 这样全球最具代表性的公司都信赖 DX。请访问 DX 官网 getdx.com/lenny。

(00:05:28) Lenny Rachitsky (Continued)

English:

Let me tell you about CommandBar. If you're like me and most users I've built product for, you probably find those little in-product pop-ups really annoying. Want to take a tour? Check out this new feature, and these pop-ups are becoming less and less effective since most users don't read what they say. They just want to close them as soon as possible, but every product builder knows that users need help to learn the ins and outs of your product. We use so many products every day and we can't possibly know the ins and outs of everyone. CommandBar is an AI-powered toolkit for product, growth, marketing and customer teams to help users get the most out of your product, without annoying them. They use AI to get closer to user intent, so they have search and chat products that let users describe what they're trying to do in their own words and then see personalized results like customer walkthroughs or actions, and they do pop-ups too, but their nudges are based on in-product behaviors like confusion or intent classification, which makes them much less annoying and much more impactful. This works for web apps, mobile apps and websites. And they work with industry-leading companies like Gusto, Freshworks, HashiCorp and LaunchDarkly. Over 15 million end-users have interacted with CommandBar. To try out CommandBar, you can sign up at commandbar.com/lenny and you can unlock an extra 1000 AI responses per month for any plan. That's commandbar.com/lenny. By the way, CommandBar just changed their name to Command AI.

中文翻译:

让我给你介绍一下 CommandBar。如果你像我以及我服务过的大多数用户一样，你可能会觉得产品里那些小弹窗非常烦人。“想参观一下吗？”“看看这个新功能”，这些弹窗的效果越来越差，因为大多数用户根本不看内容，只想尽快关掉。但每个产品构建者都知道，用户需要帮助来了解产品的方方面面。我们每天使用这么多产品，不可能知道每一个的细节。CommandBar 是一个由 AI 驱动的工具包，适用于产品、增长、营销和客户团队，帮助用户充分利用你的产品，同时不让他们感到厌烦。他们利用 AI 更贴近用户意图，提供搜索和聊天产品，让用户用自己的语言描述想做的事，然后看到个性化的结果，如客户引导或操作。他们也做弹窗，但他们的“轻推”（nudges）是基于产品内行为（如困惑或意图分类）的，这使得它们不那么烦人且更有影响力。这适用于 Web 应用、移动应用和网站。他们与 Gusto、Freshworks、HashiCorp 和 LaunchDarkly 等行业领先公司合作。超过 1500 万终端用户曾与 CommandBar 互动。要试用 CommandBar，你可以在 commandbar.com/lenny 注册，任何方案都可以每月额外解锁 1000 条 AI 回复。顺便提一下，CommandBar 刚刚改名为 Command AI。

(00:07:08) Camille Fournier

English:

The third is playing telephone. Anybody in a manager role can fall victim to this, but I think PMs especially can be very annoying. So if you are being asked questions that you cannot answer because you just don't know or because that's something that involves a level of technical detail that only the engineers have that you just don't have, and you put yourself in this in-between position where people ask you questions, you turn around, you ask the engineers questions, you take whatever they say, especially

when you don't really understand it, which happens sometimes, right? Go back to the original asker and sort of get in this middle-person scenario. I think that is very annoying and frankly, it's a waste of time for everyone. This is something that managers of all stripes do, but PMs definitely do it and that drives engineers, particularly senior engineers on projects, it's crazy.

中文翻译:

第三点是“玩传声筒游戏”（playing telephone）。任何担任管理角色的人都可能陷入这个陷阱，但我认为 PM 尤其容易让人烦。如果你被问到一些你无法回答的问题，因为你不知道，或者因为这涉及到只有工程师才掌握的技术细节，而你把自己放在了一个中间人的位置：别人问你问题，你转头去问工程师，然后把工程师说的话（尤其是当你并不真正理解时，这种情况时有发生）传达给最初的询问者。我认为这非常烦人，坦白说，这是在浪费每个人的时间。这是各种类型的管理者都会做的事，但 PM 肯定也做，这让工程师，尤其是项目中的资深工程师感到抓狂。

(00:07:51) Camille Fournier (Continued)

English:

And then, the last one that I wanted to put on this list is just when sometimes it feels like product managers want to hoard all the ideas for themselves, right? They want to be the ones that come up with every single sort of product idea and every single detail. What I see happen in those cases is that I see engineers start to over-engineer things, because engineers are like, well, I need to take control of something. I want to have some creative outlet, so I'm going to use my engineering skills as my creative outlet and I'm going to spend a lot of time obsessing over the right framework or the right this, that or the other. That may actually not matter that much with products delivery, but when you take the people that are part of the project team out of the creative loop entirely, they're going to find that creative outlet somewhere else and it's actually kind of bad for the product.

中文翻译:

最后一点是，有时感觉产品经理想要垄断所有的想法。他们想成为那个提出每一个产品创意和每一个细节的人。在这种情况下，我看到的是工程师开始“过度设计”（over-engineer），因为工程师会觉得：“好吧，我需要掌控点什么。我想要一个创意的出口，所以我要把工程技能当作我的创意出口，我会花大量时间纠结于使用哪个框架，或者这个那个技术细节。”这些对于产品交付可能并没那么重要，但当你把项目团队成员完全排除在创意环节之外时，他们会在别处寻找创意出口，而这实际上对产品是有害的。

(00:08:56) Lenny Rachitsky

English:

That's really interesting, the last one. So you're saying if you keep engineers from having a voice in what you're building and prioritizing, that's what encourages engineers to rethink, let's just rebuild this thing, let's use a new framework, let's rewrite this system.

中文翻译:

最后一点非常有意思。所以你是说，如果你不让工程师在构建什么和优先级排序上有发言权，就会促使工程师去想：“咱们把这玩意儿重造一遍吧，用个新框架，重写这个系统。”

(00:09:09) Camille Fournier

English:

Yeah, I mean that's my ... that happens without you doing that, sometimes. I do think ... I think when I see it worst and I can basically always predict what I'm going to see, a lot of that kind of engineers building stuff, finding creative outlets and kind of building stuff, maybe they shouldn't be. I'm going to find that in places where they are so quashed their creativity for the actual business or product that they're building and their voice in that is so ignored that they don't have any outlets in that space and so, they are going to use the space that they have an outlet in, the place where they have some control and that's usually the technology choices and the details there.

中文翻译:

是的，我的意思是……有时即使你不这么做，这种情况也会发生。但我认为，当我看到最糟糕的情况时，我基本上总能预见到：很多工程师在瞎搞、找创意出口、造一些可能不该造的东西，通常是因为他们在实际业务或产品上的创造力被严重压制了，他们的声音被完全忽视，以至于他们在那个领域没有出口。所以，他们会利用自己有出口、有掌控权的领域，那通常就是技术选择和技术细节。

(00:10:17) Lenny Rachitsky

English:

That's fascinating. I want to actually dig further into that around rewrites. You have a really interesting take on that, but before we get there, let me spend a little time on some of these. These are awesome. So on this theme of not involving engineers in ideation and coming up with what you're actually building, what have you seen just very tactically, is there anything you've seen that PM do super well?

中文翻译:

太有意思了。我其实想进一步探讨关于“重写”的话题，你对此有很独特的见解。但在那之前，我想在这些点上多花点时间。这些建议太棒了。关于不让工程师参与构思和决定构建内容的主题，从战术层面来看，你有没有见过 PM 做得特别好的地方？

(00:10:17) Camille Fournier

English:

I think the product managers that have done the best, they're not threatened by other people having ideas. They're not threatened by the engineering team being full of smart people because they realize that yeah, some of the engineers may have good ideas, but they still don't really know how to do the product job. Just my experience is there are plenty of engineers who actually think they can be product managers and they don't really understand all of the elements of the product job that they would need to be successful. And when product managers take the time to build those relationships, well, make sure that people do feel like they can both share their ideas, but also that they start to appreciate what the job of this product person actually is. And what they're really bringing to the table in terms of really how do we measure this input? How do we really understand the customers, how do we really think through the details of what's going to make this successful from a business or a customer perspective? I do think that that creates just a much better interaction pattern and then, engineers can feel good about sharing ideas and understanding that many of them won't go anywhere, but there's somebody that's actually going to listen and take the time to care about them.

中文翻译:

我认为做得最好的产品经理，不会因为别人有想法而感到受威胁。他们不会因为工程团队全是聪明人而感到受威胁，因为他们意识到，虽然有些工程师可能有好主意，但他们仍然不知道如何做产品工作。根据我的经验，有很多工程师觉得自己能当 PM，但他们并不真正理解成功完成产品工作所需的全部要素。当 PM 花时间建立好

这些关系，确保大家觉得既可以分享想法，又开始欣赏这个产品人的工作价值时——比如他们如何衡量投入？如何真正理解客户？如何从业务或客户角度思考成功的细节？我认为这会创造出更好的互动模式。这样工程师在分享想法时会感觉良好，即使知道其中很多想法不会被采纳，但至少有人在倾听，并愿意花时间关注他们。

(00:11:37) Lenny Rachitsky

English:

Coming back to some of the things that you said annoy engineers of PMs, this idea of playing the middle person, sounds like the solution clearly is there. Just connect the engineer to the other engineer or your engineer to the PM that's trying to figure this out, right?

中文翻译:

回到你提到的 PM 让工程师讨厌的那些事，关于“充当中间人”这一点，听起来解决方案很明确：直接让工程师对接另一个工程师，或者让你的工程师对接那个想弄清楚问题的 PM，对吧？

(00:11:49) Camille Fournier

English:

That one is not always easy because again, you have this tension of a lot of the job, of any management role is being in meetings and filtering stuff so that people who are in focus ... individual contributor mode can focus and get things done and not spend half of their weekend in meetings. You've just got to be very careful about knowing when you're crossing that line and when you're crossing it too often. And if you're having to often say, "Let me get back to you, let me get back to you, I don't know, let me get back to you." Maybe the person you're talking to is asking the wrong level of questions of you. Maybe you need to connect them to the engineers directly, but just being aware that that shouldn't be a default behavior. That will happen occasionally, but it shouldn't be a thing that happens a lot because if it's happening a lot, then you're likely missing something, you're likely losing something in that telephone game translation and that's going to cause problems over time.

中文翻译:

这并不总是那么容易，因为这里存在一种张力：任何管理角色的很大一部分工作就是开会和过滤信息，以便处于专注模式的个人贡献者（IC）能够集中精力完成工作，而不是把周末的一半时间花在开会上。你必须非常小心，知道什么时候越界了，以及是否越界太频繁。如果你不得不经常说“我晚点回复你，我不知道，我晚点回复你”，也许和你谈话的人问的问题层级不对。也许你需要让他们直接对接工程师，但要意识到这不应该成为默认行为。偶尔发生没关系，但不应该经常发生，因为如果经常发生，你可能遗漏了什么，或者在“传声筒游戏”的翻译中丢失了信息，久而久之会出问题。

(00:12:47) Lenny Rachitsky

English:

Awesome. So basically, if you're just finding your middle person too much, then it may be time to connect people directly. And I know the reason PMs often are afraid of this is the engineer may agree to something that they think is a bad idea for their team or may not understand all the ramifications on the product or just obviously, just spend their time in meetings and not be building anything.

中文翻译:

太棒了。所以基本上，如果你发现自己充当中间人的次数太多，那就是时候让人们直接沟通了。我知道 PM 经常害怕这一点的原因是：工程师可能会答应一些 PM 认为对团队不利的事情，或者工程师可能不理解对产品的全部影响，或者显而易见地，工程师会把时间都花在开会上而没法写代码。

(00:13:12) Camille Fournier

English:

Yeah, yeah, and sometimes you mean you do it in a group meeting. I feel like Slack and other chat type things actually make it a lot easier to see, have the right people in a group in a thing, but again, that's distracting. So there's not an easy solution to that one, just I think it's important to be aware of it.

中文翻译:

没错，有时你可以通过小组会议来解决。我觉得 Slack 和其他聊天工具实际上让拉对人进群组变得容易多了，但同样，这也会分散注意力。所以这没有简单的解决方案，我认为意识到这一点很重要。

(00:13:27) Lenny Rachitsky

English:

Yeah, that's a really good point. And then, in terms of hoarding credit, is there any tactical thing you've seen PMs do really well here is it, just every time they're announcing the product, "Hey, these engineers were involved or-"

中文翻译:

是的，很有道理。关于“独揽功劳”，你有没有见过 PM 在这方面做得特别好的战术动作？是每次发布产品时都说“嘿，这些工程师也参与了”还是……？

(00:13:37) Camille Fournier

English:

It's more than just saying thank you to all these people, but it's actually sometimes stepping back and letting other people speak, especially if it's something that's a really, really big technical lift. I don't think there's a super easy fix to that. I think it is just really being mindful that that can be very much a sore point for engineers when they just feel like, "This is my work, I'm not getting any credit for it, and this person is hogging all the glory."

中文翻译:

不仅仅是口头感谢这些人，有时实际上是退后一步，让其他人发言，特别是当涉及到非常重大的技术提升时。我不认为这有超级简单的解决方法。我认为关键是要意识到，当工程师觉得“这是我的工作，我却没得到任何认可，而这个人却独占了所有光环”时，这会成为他们的痛点。

(00:14:07) Lenny Rachitsky

English:

I love that. Yeah, I find the best PMs are the ones that talk the least and encourage other people to do the presenting and announcing. And so, I think that's a really good reminder is let your engineers do that. Okay, amazing. So we talked a little bit about this idea of rewriting and how engineers sometimes just

want to rewrite the system and I think a lot of PMs do too. A lot of times you're building your features on a thing that someone that doesn't even work at the company and were built five, 10 years ago, and there's always this sense of, "Okay, maybe we should just rewrite this thing everything will move so much faster." Do you have a really interesting take that I think a lot of PMs will love to hear, which is that rewrites are often a big trap and often don't end up being what you think they might be? Can you just talk about your experience and perspective on this?

中文翻译:

我喜欢这个观点。是的，我发现最好的 PM 往往是那些话最少，并鼓励其他人进行演示和发布的人。所以我觉得这是一个很好的提醒：让你的工程师去做这些。太棒了。我们刚才聊到了“重写”的想法，工程师有时就是想重写系统，我觉得很多 PM 也有这种想法。很多时候，你是在一个五年前、十年前由已经不在公司的人构建的东西上开发功能，总会有一种感觉：“好吧，也许我们应该重写这玩意儿，这样一切都会快得多。”你有一个非常有趣的观点，我觉得很多 PM 会爱听，那就是：重写往往是一个巨大的陷阱，而且结果往往不如你所愿。你能谈谈你在这方面的经验和看法吗？

(00:14:50) Camille Fournier

English:

Yeah, so I mean I have personally overseen a number of, if not quite rewrites, re-architectures and major system evolutions. So I absolutely do think they are sometimes a thing that needs to happen, but I also, have seen so many instances of cases where the engineers have convinced themselves that the only solution to the woes that they're experiencing with the system, it's hard to support, it's hard to change. Nobody wants to work on it, because it's this old crappy technology, is that they just have to go over to the side, build the new thing that will replace this old system and that is going to sort of free them from their misery.

中文翻译:

是的，我个人监督过很多次——如果算不上完全重写的话——架构重构和重大系统演进。所以我绝对认为它们有时是必须发生的。但我也见过太多这样的情况：工程师们说服自己，解决系统痛苦（难以维护、难以更改、没人想碰，因为技术又老又烂）的唯一办法就是躲到一边，构建一个能替代旧系统的新东西，从而把他们从痛苦中解救出来。

(00:15:41) Camille Fournier (Continued)

English:

And I think projects where you acknowledge that you do need to do an uplift, but you make a very thoughtful staged plan as to how you're going to do that, and you really think through, okay, we don't need to touch all of this stuff, but we're going to take the recommendation system, it really needs to be uplifted and that's a well-contained, you know, API and so we can start to fix that without having to change the whole whatever web framework, right? So I do think there are ways to do these evolutions, but people really underestimate. They underestimate the time to migrate stuff from the old system to the new system, is a huge, huge problem. Particularly when you're talking about systems where you have sort of external people using the system in some way, whether it's web UIs or APIs. You think, "Oh, we kind of know what's going on, so it's not going to be that big a deal." Engineers notoriously, notoriously, notoriously, massively underestimate the migration time for old system to new system and that causes a lot of problems.

中文翻译:

我认为，在那些你承认确实需要提升，但制定了非常周密的阶段性计划的项目中，你会仔细思考：好吧，我们不需要动所有的东西，但我们要拿推荐系统开刀，它确实需要提升，而且它是一个封装良好的 API，所以我们可以开始修复它，而不需要更改整个 Web 框架，对吧？所以我认为有办法进行这些演进，但人们真的低估了难度。他们低估了将东西从旧系统迁移到新系统的时间，这是一个巨大的问题。特别是当你谈论那些有外部人员以某种方式使用的系统时，无论是 Web UI 还是 API。你觉得“哦，我们大概知道是怎么回事，所以没那么难”。工程师们极其、极其、极其严重地低估了从旧系统到新系统的迁移时间，这导致了很多问题。

(00:16:24) Camille Fournier (Continued)

English:

By the way, you still have to support the old system while you're working on the new system. So I doubt many of the PMs in the audience are ever happy when they hear, we need to go away for six months, a year, two years to build this new thing and we just can't really add any features to the system in the interim. That's infuriating, I'm sure, and frankly that's a problem and I don't know that that should be an acceptable answer in many cases. There may occasionally again be a case where that is what has to happen, but I think most of the time you can't really afford to just say we're going to go away and we're not going to touch the system for a long time and we're going to build something new over here.

中文翻译:

顺便说一句，在开发新系统的同时，你仍然必须支持旧系统。所以我怀疑在座的 PM 听到“我们需要闭关六个月、一年、两年去造这个新东西，期间我们没法给系统增加任何功能”时，会感到高兴。我确信这让人抓狂，坦白说这就是个问题，而且我不认为在很多情况下这应该是一个可以接受的答案。偶尔可能确实必须如此，但我认为大多数时候你承担不起“我们要消失很久，不碰旧系统，在这里造个新东西”的代价。

(00:17:47) Camille Fournier (Continued)

English:

So many things about why that doesn't make any sense. So this is a little bit of field of the blog post that you're referring to. So, if you've got a system that doesn't really need feature enhancement or development because it's just sort of fine and the users are using it. And it's just annoying to the engineers, why in the world would you invest so much money in writing a new version of it? There's a little bit of a cognitive dissonance that sometimes happens if you need to do new stuff and the old system literally is not ... it's not possible to do the new stuff that you need to do, you need to figure out a path to get to a sustainable system or you can continue to add and evolve. You should be investing and so there does need to be an investment, but you have to ask yourself, if I could go away and not touch this and not do anything to it for a long period of time without it really harming my business, is it worthwhile to change it at all?

中文翻译:

有很多理由说明为什么这行不通。这涉及到你提到的那篇博文的内容。如果你的系统并不真正需要功能增强或开发，因为它运行得还行，用户也在用，只是让工程师觉得烦，那你到底为什么要投入这么多钱去写个新版本？有时会发生认知失调：如果你需要做新功能，而旧系统确实无法实现，你需要找到一条通往可持续系统的路径，以便继续添加和演进。你应该进行投资，所以确实需要投入，但你必须问自己：如果我可以很长时间不碰它、不对它做任何事，而这并不会真正损害我的业务，那么改变它到底值不值得？

(00:18:28) Camille Fournier (Continued)

English:

Does it matter, and there's some questions there. I also think that when people try to do rewrites, particularly again if it's something that you're really trying to just move to a new language for example or sort of modernize in a certain way, [inaudible 00:18:45] a lot of times people really underestimate what the old system does and how well they know what the old system does. There's so much logic buried in legacy systems, it tends to be undocumented, it tends to be weird. You haven't thought through all the business rules, you haven't thought through the data formatting and I think again, it's much, much harder to replicate all the important things from the old system to the new system than people expect. So there's more than that, but I do think sometimes you need to evolve systems and my advice would be when you're struggling making an evolution plan, take pieces potentially of the old system, uplift them, make them more scalable, make them easier to work with, clean up the tech debt, but trying to say we're going to just go away. We're going to rewrite, we're going to build something brand new and it's going to solve all our problems, it just very rarely works.

中文翻译:

这很重要吗？这里有一些疑问。我还认为，当人们尝试重写时，特别是如果你只是想换一种语言或者以某种方式实现现代化，很多时候人们严重低估了旧系统的功能，以及他们对旧系统的了解程度。遗留系统中埋藏了太多的逻辑，往往没有文档，往往很古怪。你没有想清楚所有的业务规则，没有想清楚数据格式，我认为要将旧系统中所有重要的东西复制到新系统中，比人们预想的要难得多。不仅如此，我确实认为有时你需要演进系统，我的建议是：当你纠结于制定演进计划时，尝试把旧系统拆解，提升其中的一部分，让它们更具扩展性，更容易使用，清理技术债，但如果想说“我们要消失一段时间去重写，造个全新的东西来解决所有问题”，这极少能成功。

(00:19:42) Lenny Rachitsky

English:

I think a lot of PMs will be like, "Yes, thank you so much for saying this, because I think that's also always a big struggle between the ENG team and the PM team." So just to summarize what I think a lot of people miss or what you're saying a lot of people miss when they're thinking about let's rewrite this thing, is the migration to the new system, migrating customers, users, data to the new thing is going to take a lot longer than you expect. You underestimate knowing what it actually does and you're going to miss features and you can introduce new bugs. This actually is very similar to what I've seen with redesigning a whole product flow. There's always this sense of let's just rethink this onboarding flow from scratch or let's rebuild this part of the product and always it ends up being a negative experiment result. It always ends up being less good and then, you have to spend all this time clawing back to get to where you were and also, you forget the stuff that would ... the features that you had and you're like, "Oh, shit, I forgot about that feature, I forgot about that feature." So it's interesting, there's a very similar situation in the product side.

中文翻译:

我想很多 PM 会说：“太好了，非常感谢你这么说，因为这总是工程团队和 PM 团队之间的一大矛盾点。”总结一下你认为很多人在考虑重写时忽视的点：迁移到新系统、迁移客户、用户和数据的时间会比预想的长得多。你低估了了解系统实际功能的难度，你会遗漏功能，还会引入新 Bug。这实际上和我见过的重新设计整个产品流程非常相似。总有一种感觉：“让我们从头开始重新思考这个入职流程吧”或者“让我们重建这部分产品”，结果往往是负面的实验结果。它总是变得更糟，然后你不得不花大量时间退回到原来的状态，而且你还会忘记原来的功能，然后惊呼：“噢，该死，我忘了那个功能了。”所以很有趣，产品端也有非常类似的情况。

(00:20:45) Lenny Rachitsky (Continued)

English:

Okay, amazing. I'm going to go to a slightly different topic, which is around engineering leadership. So I know you've written a lot about engineering leadership, you spent a lot of times with engineering leaders, so I have a few questions here. One is that I know that one of the things that haunts engineering leaders most is finding the balance between staying technical and their technical expertise, and their leadership expertise and basically finding the right altitude of how high ... where to be in the org and also how in the details to be, and also staying technical enough to be relevant. What have you learned in your own experience of finding that balance and how do you advise engineering leaders as they struggle with this?

中文翻译:

好，太棒了。我要转到一个稍微不同的话题，关于工程领导力。我知道你写过很多关于工程领导力的文章，也花了很多时间与工程领导者交流，所以我这里有几个问题。第一，我知道最困扰工程领导者的事情之一就是：如何在保持技术能力（技术专长）和领导力专长之间找到平衡。基本上就是找到合适的“高度”——在组织中处于什么位置，对细节介入到什么程度，以及如何保持足够的技术能力以不落伍。在你寻找这种平衡的经验中，你学到了什么？你如何建议那些为此挣扎的工程领导者？

(00:21:18) Camille Fournier

English:

Yeah, so one piece of advice I give everybody is don't stop being a hands-on technical until you feel like it's in your bones. You feel like you've got mastery that you could ... if you know a second language fluently or if you played an instrument really, really seriously for a long time or maybe a sport really, really seriously for a long time, you'll be familiar with the ... I haven't done that in a long time, but if I was to pick it up, it would be rusty, but I would get there pretty quickly, right? Maybe physically, I wouldn't be as strong as I was or whatever, but I would get there. You can do that with writing code. You can do that with technical skills if you do it for long enough, I think you can develop sort of a baseline mastery, where you're not going to be as fast and a lot of the challenges of being technical is actually in all the tooling and all the tooling evolution, but you're not going to be necessarily as fast as people who have been doing it, but you won't be completely clueless and I think that all the things you learn getting to that really comfortable mastery of some part of hands-on tech will stay with you and will help you just maintain a level of confidence in your own technical know-how and maintain a level of empathy for what it means to be a good engineer.

中文翻译:

是的，我给每个人的一个建议是：在技术还没进入你的“骨髓”之前，不要停止亲自动手做技术。你要感觉到自己已经掌握到了这种程度……就像如果你流利地掌握了第二语言，或者长期认真地演奏某种乐器，或者长期认真地从事某种运动，你会熟悉这种感觉：我很久没做了，如果我现在重新开始，会有些生疏，但我能很快找回感觉，对吧？也许体力上不如以前强壮，但我能做到。写代码也是如此。如果你做技术的时间足够长，你可以培养出一种基础的精通感。虽然你可能没那么快（做技术的挑战很大程度上在于工具及其演进），你未必能像那些一直在一线的人那么快，但你不会完全一窍不通。我认为，在达到对某部分动手技术的舒适精通过程中学到的所有东西都会伴随你，帮助你保持对自身技术诀窍的信心，并保持对“什么是优秀工程师”的同理心。

(00:22:38) Camille Fournier (Continued)

English:

And I think just make you a lot less anxious about being hands-off, even though I think everybody who makes that transition for a year or two, especially if you're really have to be hands-off or you just don't have time to write code at all, you are going to be anxious for a while no matter what. The things to then think about from that point is, being technical is also just about knowing what's going on and paying attention and being able to ask ... what people care about with technical leaders in my experience is they want people who actually seem like they sort of understand what you're doing and can ask good questions and help guide you to better decisions without actually being the one who's like, "Oh no, you need to use this library instead of that library." It's actually sort of annoying when somebody that's very senior and hands-off tries to tell you, don't use this library, use that library because I don't know about you, but I don't really believe people who have been hands-off for that long when they try to tell me what to do and the thing that I'm kind of the expert in right now, but I do appreciate it when I'm given more guidance around, well, have you considered this? Tell me about how you're planning to handle that situation. What are the major technical challenges with implementing this and that can actually spend the time to listen and ask thoughtful questions on the back of that.

中文翻译:

我认为这会让你在不再亲自动手时少一些焦虑，尽管我认为每个经历这种转型一两年的人，特别是如果你真的必须放手或者根本没时间写代码，无论如何你都会焦虑一段时间。从那时起要考虑的是：保持技术性也意味着了解正在发生的事情，保持关注并能够提问。根据我的经验，人们对技术领导者的期望是，他们希望领导者看起来真的懂你在做什么，能提出好问题，并引导你做出更好的决定，而不是亲自跳出来说：“噢不，你应该用这个库而不是那个库。”当一个非常资深且不亲自动手的人试图告诉你“别用这个库，用那个库”时，其实挺烦人的。因为我不知道你怎么想，但我不怎么相信那些脱离一线很久的人在他们并不精通的领域对我指手画脚。但我很感激他们能给我更多引导，比如：“你考虑过这个吗？”“告诉我你打算如何处理那种情况？”“实现这个的主要技术挑战是什么？”并且能真正花时间倾听并在此基础上提出深刻的问题。

(00:24:00) Camille Fournier (Continued)

English:

The last thing I would say is surround yourself with smart technical people, also as much as you can, and be willing to listen to them, talk about tech and ask them questions about things. I feel like that's part of the reason that I am able to stay technically savvy and credible amongst people who work for me is not that I'm writing code because I'm not. But I am listening to a lot of very smart people talk about technology a lot, down to the level of I'm trying to debug this database issue, what the heck is going on? And just constantly being interested in those stories and learning from them and learning what really smart engineers are thinking about and worrying about. The more you can build that network of people that are still hands-on and stay in touch with that, I do think that helps a lot.

中文翻译:

最后我想说的是，尽可能让自己周围都是聪明的技术人才，并愿意听他们谈论技术，向他们请教。我觉得这就是我能在下属面前保持技术敏锐度和可信度的部分原因——并不是因为我在写代码，因为我并没写。但我经常听很多非常聪明的人谈论技术，细致到“我正在调试这个数据库问题，到底是怎么回事？”我一直对这些故事感兴趣，从中学习，了解那些真正聪明的工程师在思考什么、担心什么。你越能建立起这样一个仍在一线的人脉网并保持联系，我认为帮助就越大。

(00:24:56) Lenny Rachitsky

English:

So on my last point, how are you actually doing that? Is it watching ... going to conferences with friends, something else?

中文翻译:

关于最后一点，你具体是怎么做的？是看……和朋友一起参加会议，还是别的什么？

(00:25:01) Camille Fournier

English:

Yeah, yeah. I mean, I guess for me it started with going to conferences, meeting people. Now, I'm in a lot of different chat groups where people are just sort of regularly communicating, staying in touch with ... staying in touch with former colleagues. I will admit I'm kind of a social person and I have a big network, so this may be easier said than done, but I do think being in the right group chats ... I also think, I'm sure reading various tech news and tech sort of commentary and discussion boards, I mean it's definitely a mixed bag of that stuff I think that like ... but there are smart people. You find the smart people in there, you sort of follow what they're saying. I think that's another good way to keep that perspective.

中文翻译:

是的。对我来说，起初是参加会议、结识朋友。现在，我在很多不同的聊天群里，大家会定期交流，和前同事保持联系。我承认我比较社交化，人脉很广，所以这可能说起来容易做起来难。但我确实认为加入正确的群聊很有帮助。我也确信阅读各种技术新闻、技术评论和讨论版块也有用，虽然那些东西鱼龙混杂，但里面有聪明人。你在里面找到聪明人，关注他们的言论。我认为这是保持视角的另一种好方法。

(00:25:51) Lenny Rachitsky

English:

Is it a sign, I wonder if you're not interested in that anymore that maybe you should move into something else. I don't know. If you're like, don't really pay attention to engineering technicalness.

中文翻译:

我在想，如果你不再对这些感兴趣了，是不是一个信号，说明也许你应该转行做别的了？我不知道。如果你不再关注工程技术性的话。

(00:26:02) Camille Fournier

English:

It's hard for me to say because I'm such a nerd. I really love tech. I'm in this industry, because I'm just actually genuinely very interested in certain corners, not every corner of technology but certain corners of technology. I want to know what the latest stuff that's happening in databases and infrastructure and I find it all very interesting. I find the problems interesting. So I think that makes me very successful because I just have that natural curiosity and passion and interest in it. But I don't know that that's a total prerequisite.

中文翻译:

这对我来说很难说，因为我就是个“技术宅”（nerd）。我真的很爱技术。我进入这个行业是因为我真的对技术的某些领域（不是所有领域，而是某些领域）非常感兴趣。我想知道数据库和基础设施领域的最新动态，我觉

得这些都很有趣。我觉得这些问题很有趣。所以我认为这让我非常成功，因为我对此有天然的好奇心、激情和兴趣。但我不知道这是否是一个绝对的前提条件。

(00:26:48) Lenny Rachitsky

English:

Yeah, but you've been talking about ... it reminds me a little bit of this guy, LevelsIO on Twitter. Have you heard of this guy? He was just on Lex Friedman. Have you listened to that yet?

中文翻译:

是的，但你刚才说的……让我想起了 Twitter 上的那个叫 LevelsIO 的家伙。你听说过他吗？他刚上了 Lex Friedman 的播客。你听了吗？

(00:26:47) Camille Fournier

English:

I think maybe I saw a clip of it, but I haven't listened to it.

中文翻译:

我想我可能看过一个片段，但还没听过。

(00:26:48) Lenny Rachitsky

English:

So I think one of the most successful indie engineers where he just works on his own thing all by himself, never raises money, just launch his products that make money. And a funny thing about him is he works ... all stuff is in PHP and jQuery. He's just like, this works. I've always had this to do, learn Node.js, learn Python. And I'm like, I'm too busy to build ... while I'm building to learn these new things and he's been incredibly successful. So it touches on sometimes maybe you don't need to just keep rewriting to the newest frameworks.

中文翻译:

我认为他是最成功的独立工程师之一，他一个人单打独斗，从不融资，只发布赚钱的产品。关于他有一件很有趣的事：他所有的东西都是用 PHP 和 jQuery 写的。他觉得“这能行”。他总说“我有计划学 Node.js，学 Python”，但又说“我忙着构建产品，没时间学这些新东西”，而他非常成功。所以这触及了一个点：有时也许你不需要一直为了追求最新框架而重写。

(00:27:15) Camille Fournier

English:

Yeah, no, I mean look, I actually ... I feel like I know a lot of smart engineers who are in that category. They're like, we built amazing things in PHP and relatively simple SQL and so much of tech is over-engineering things and I don't totally disagree. I think the challenge is though, of course, what works as a one person show, doesn't always work in a scaled organization for better or for worse, we're in different like, you've got to match what makes one person really productive. Will it make 100 people, 1000 people, even 10 people really productive? That's always a little hard to tell and that's why I do think you should

be not ... I think it's always a good idea to be keeping up with what's happening and what's changing in whatever kind of side of tech you're in, but not obsessively chasing every fad. I think being aware of, but not necessarily chasing them, but particularly, if you're working in groups, teams, larger companies, even midsize companies, there is some amount of, you're balancing the tech that makes one person go fast with the tech that makes 10 or 100 people go fast and those are not always exactly the same thing.

中文翻译:

是的，没错。其实我觉得我认识很多属于这一类的聪明工程师。他们会说：“我们用 PHP 和相对简单的 SQL 构建了了不起的东西。”很多技术确实存在过度设计的问题，我并不完全反对这个观点。但挑战在于，无论好坏，一个人单打独斗行得通的东西，在规模化组织中并不总是行得通。你必须匹配让一个人高效的东西，是否也能让 100 人、1000 人甚至 10 人高效？这总是很难说。这就是为什么我认为你不应该……我认为紧跟技术领域的变化总是个好主意，但不要盲目追求每一个潮流。我认为要保持关注，但不一定要追逐。特别是如果你在团队、大公司甚至中型公司工作，你需要在“让一个人跑得快的技术”和“让 10 人或 100 人跑得快的技术”之间做平衡，而这两者并不总是一回事。

(00:28:35) Lenny Rachitsky

English:

Just to kind follow this thread a little bit and to kind of nerd snipe you a little bit, is there a platform or language or framework these days that you're either very excited about that you think is helping people move faster and do better work or the opposite just like this is ... everyone is excited but this is not good.

中文翻译:

顺着这个话题，我想“技术性偷袭”（nerd snipe）你一下：最近有没有什么平台、语言或框架让你非常兴奋，觉得能帮人跑得更快、做得更好？或者相反，大家都觉得很兴奋但你觉得并不好？

(00:28:52) Camille Fournier

English:

I will say this, one example I actually put in my own notes for this conversation was GraphQL. I would not tell a team to use or not use GraphQL at this point because it's a bit out of my expertise zone and my level of management. It's not really my job anyway, but it is one of the things that is both popular and thought relatively poorly of by most of the senior people that I know. And so, I guess I would say that's one where I would say if you're seriously thinking about it and you're not Facebook, you may really want to make sure you know what problem you're trying to solve because the impression that I have from sort of listening to people talk about it is that GraphQL is kind of trying to promise front-end engineers that they don't really have to collaborate with backend engineers. And they can just sort of build whatever and it'll all be fine, and it just doesn't ever seem to work out that well for anybody who actually does it in practice. Again, obviously, it can work out that well because Facebook has made a great go of it. I'm sure there are other companies that are, but that's one where it's not that new but it remains one of these things where it seems like an interesting fad that maybe is burning a lot of people.

中文翻译:

我会这么说，我为这次谈话准备的笔记中有一个例子就是 GraphQL。目前我不会告诉一个团队该不该用 GraphQL，因为这有点超出我的专业领域和管理层级，反正也不是我的职责。但它是那种既流行，又被我认识的大多数资深人士评价相对较低的东西。所以我想说，如果你在认真考虑它而你又不是 Facebook，你可能真的需要确保你知道自己想解决什么问题。因为我听人谈论它的印象是：GraphQL 似乎在向前端工程师承诺，他们不需要真正与后端工程师协作，可以随心所欲地构建，一切都会没问题。但在实践中，这对大多数人来说似乎

效果并不好。当然，显然它也可以运行得很好，因为 Facebook 做得很成功，肯定还有其他公司也是。但它属于那种虽然不新，但仍然像是一个让很多人“踩坑”的有趣潮流。

(00:30:13) Lenny Rachitsky

English:

That's awesome. I appreciate you sharing that. I don't know if this is exactly an example of this, but on that podcast levels, I forget his actual name, Peter I think, shared that whenever there's a framework that has a VC funded startup behind it, that's not a good sign because their job now is to convince engineers to use it and then pay for it and that's not necessarily going to be the best product.

中文翻译:

太棒了。感谢分享。我不知道这是否是一个恰当的例子，但在 Levels 的那个播客中（我忘了他的真名，好像叫 Peter），他分享说：每当一个框架背后有一家风投支持的初创公司时，这通常不是个好兆头。因为他们的工作现在是说服工程师使用它，然后为此付费，而这并不一定意味着它是最好的产品。

(00:30:34) Camille Fournier

English:

That's true. Although I will say that some of the most time-wasting frameworks have also just come out of big companies, or the context of the big company that may have made that framework super useful within that context doesn't translate to startup or small company or even big company that doesn't have the rest of the context set, but I don't think ... He's not wrong. I also just think big companies share the blame on that one.

中文翻译:

没错。虽然我想说，一些最浪费时间的框架也出自大公司。大公司的背景可能让那个框架在特定环境下超级好用，但这种背景无法转化到初创公司、小公司，甚至是没有相同背景的其他大公司。但他没说错，我也觉得大公司在这方面也难辞其咎。

(00:31:05) Lenny Rachitsky

English:

I guess we should all just come back to PHP and jQuery and be simple.

中文翻译:

我想我们都应该回归 PHP 和 jQuery，保持简单。

(00:31:09) Camille Fournier

English:

Maybe.

中文翻译:

也许吧。

(00:31:10) Lenny Rachitsky

English:

Okay, so to close out this thread on engineering leaders, finding the right balance, just to summarize your advice here. One is get to mastery before and is your advice here, get to this point before you move into engineering.

中文翻译:

好，为了结束关于工程领导者寻找平衡的话题，总结一下你的建议：第一是在转型之前达到精通。你的建议是，在进入工程管理之前要达到这个点。

(00:31:23) Camille Fournier

English:

Engineering management. I will say part of this is also in particular, if you happen to be a woman or otherwise, underrepresented person in tech because people will tend to underestimate your technical abilities just unfortunately as a get-go. I also think it's particularly important to kind of develop that internal confidence in your abilities before you make this sort of scary leap, which is scary for everyone of have a ... if you have that mastery before you make the leap, I wish more people would do this because honestly, I do think there are a lot of people who never really gained the mastery. They go into management, they lose it and some of them are still perfectly good managers and look, there are good managers who were never technical to begin with. I don't want to say that that's impossible. I just think that if you care about being technical, if you are technical now and you want to maintain that tech-savvy, don't just become a manager the first time somebody offers it to you. Make sure you've really spent your good time writing code.

中文翻译:

是工程管理。我想说，特别是如果你恰好是女性或科技界其他少数群体，不幸的是，人们往往从一开始就会低估你的技术能力。我认为在迈出这可怕的一步（对每个人来说都很可怕）之前，培养对自己能力的内在信心尤为重要。如果你在跳槽前已经达到了精通，我希望更多人能这样做。因为坦白说，我认为有很多人从未真正达到精通。他们进入管理层，然后失去了技术能力，其中一些人仍然是完美的管理者。听着，确实有从一开始就不是技术出身的好管理者，我不想说那是不可能的。我只是觉得，如果你在乎技术，如果你现在是技术人员并想保持技术敏锐度，不要在别人第一次邀请你当经理时就答应。确保你真的花了足够的时间写代码。

(00:32:28) Lenny Rachitsky

English:

Is there a number of years heuristic you think about or some way to tell you that maybe you've hit that point?

中文翻译:

你有没有一个关于年限的启发式标准，或者某种方法能告诉你可能已经达到了那个点？

(00:32:34) Camille Fournier

English:

I think this has been since disproven, but it was that 10,000 hours idea of mastery at some point. For me, it was like an undergraduate degree, a graduate degree, and four or five years of full-time work. So maybe I might be slow. I didn't start coding a lot in middle school like people might do now, but I felt like ... I took several years of hands-on work in a very intense undergraduate and graduate programs for me. So I do think it's probably somewhere in the 10-year range of really having spent a lot of your time over those years writing code and really understanding how to be a technical expert.

中文翻译:

虽然“一万小时定律”后来被证伪了，但它代表了某种精通的概念。对我来说，是本科学位、研究生学位加上四五年的全职工作。所以我可能比较慢。我没有像现在的孩子那样在中学就开始大量写代码，但我觉得……我经历了非常高强度的本科和研究生课程，再加上几年的动手工作。所以我认为可能在 10 年左右的范围内，你真的在这些年里花了大量时间写代码，并真正理解了如何成为一名技术专家。

(00:33:17) Lenny Rachitsky

English:

Got it. So essentially if you're thinking about moving into management as an engineer, you may want to wait until you've done it for 10 years in some form, which I think is a lot longer than a lot of people would've thought. And I imagine many people are not doing that. And then, in your experience not doing it as well, it could be-

中文翻译:

明白了。所以基本上，如果你考虑作为工程师转入管理层，你可能需要等待，直到你以某种形式做了 10 年技术。我觉得这比很多人预想的要长得多。我猜很多人并没有这样做。根据你的经验，没做到这一点可能会……

(00:33:35) Camille Fournier

English:

If you were programming a lot in high school and you got an undergraduate degree, so let's say you've got six years-

中文翻译:

如果你在高中就开始大量编程，然后拿到了本科学位，那么算你有 6 年经验——

(00:33:42) Lenny Rachitsky

English:

I see.

中文翻译:

我明白了。

(00:33:42) Camille Fournier

English:

You may only need four or five years of work, 40 hour a week writing code experience. I'm sure it depends on the company, but I do think when you see people that are 23, they are just out very recently out of school, it's a different thing if you're a founder and that's a whole different life, but if you're at a big company and somebody is like you have great communication skills, why don't you start to become a manager? Often they're actually pushing you to become a project manager, which is actually also the worst sort of path to real leadership in my opinion. If you don't feel like you're done ... Also, if you just don't feel like you're done, if you're still having fun writing code, don't rush becoming a manager, writing code is awesome. Have fun, enjoy it.

中文翻译:

你可能只需要四五年的工作经验，即每周 40 小时写代码的经验。当然这取决于公司，但我确实认为，当你看到 23 岁、刚走出校门的人时——如果你是创始人那是另一回事，那是完全不同的人生——但如果你在大公司，有人说“你沟通能力很强，为什么不开始当经理呢？”，通常他们实际上是在推你去做项目经理（Project Manager），在我看来，那是通往真正领导力最糟糕的路径。如果你觉得还没写够……而且，如果你觉得写代码还有趣，就不要急着当经理。写代码很棒，享受它。

(00:34:29) Lenny Rachitsky

English:

I know exactly what you mean. So I used to be an engineer, actually I was an engineer for 10 years. I definitely don't have mastery at this point. I moved into product from that and I definitely so missed actually just sitting there and writing code and building stuff, that was very hard to give up. I imagine you still missed that.

中文翻译:

我完全明白你的意思。我以前也是工程师，实际上我做了 10 年工程师。我现在肯定谈不上精通。我后来转到了产品岗位，我真的非常怀念那种坐在那里写代码、构建东西的感觉，放弃它非常难。我想你现在仍然怀念。

(00:34:45) Camille Fournier

English:

I think I might've forgotten about it at this point, but there is nothing as satisfying because you get the fast feedback loop. It's just wonderful. Yeah.

中文翻译:

我想我现在可能已经忘了那种感觉了，但没有什么比写代码更让人满足的了，因为你能得到快速的反馈循环。那感觉太棒了。是的。

(00:34:56) Lenny Rachitsky

English:

This episode is brought to you by Coda. I use Coda every day to coordinate my podcasting and newsletter workflows from collecting questions for guests to storing all my research to managing my newsletter content calendar, Coda is my go-to app and has been for years. Coda combines the best of documents, spreadsheets and apps to help me get more done and Coda can help your team to stay aligned and ship faster by managing your planning cycle in just one location. Set and measure OKRs with full visibility across teams and stakeholders, map dependencies, create progress visualizations, and identify risk areas.

You can also access hundreds of pressure tested templates for everything from roadmap strategy, to final decision-making frameworks. See for yourself why companies like DoorDash, Figma and Qualtrics run on Coda. Take advantage of this special limited-time offer just for startups. Head over to coda.io/lenny and sign up to get six free months of the team plan. That's coda.io/lenny to sign up and get six months of the team plan, coda.io/lenny.

中文翻译:

本集节目由 Coda 赞助。我每天都使用 Coda 来协调我的播客和时事通讯工作流，从收集嘉宾问题到存储所有研究资料，再到管理时事通讯内容日历，Coda 多年来一直是我的首选应用。Coda 结合了文档、电子表格和应用的优点，帮助我完成更多工作。Coda 还可以通过在一个地方管理规划周期，帮助你的团队保持一致并更快交付。设定并衡量 OKR，在团队和利益相关者之间实现完全透明，映射依赖关系，创建进度可视化，并识别风险区域。你还可以访问数百个经过压力测试的模板，涵盖从路线图策略到最终决策框架的所有内容。亲自去看看为什么像 DoorDash、Figma 和 Qualtrics 这样的公司都在 Coda 上运行。利用这个专门为初创公司提供的限时特别优惠。前往 coda.io/lenny 注册，即可免费获得 6 个月的团队计划。

(00:36:24) Lenny Rachitsky

English:

On the topic of moving into management, you wrote maybe the definitive book on engineering manager career path, and so when someone moves from IC to management, what do you find is the most surprising thing to them? What do they most often not understand or are surprised by like, "Oh man, I did not see this as part of my job or my life."

中文翻译:

关于转型管理的话题，你写了那本可能是关于工程经理职业路径的权威著作。那么当有人从 IC 转型为管理时，你发现最让他们感到意外的是什么？他们最常不理解或感到惊讶的是什么？比如：“天哪，我没预料到这也是我工作或生活的一部分。”

(00:36:24) Camille Fournier

English:

Yeah, I mean I think there's a few things. I do think ... assuming that they're actually trying to do it well, I do think there are a lot of people who move into management and then just don't really understand the job at all and aren't even self-aware enough to know that they don't understand it, but for those who are trying to do it, trying to do it well. I think a few things that tend to surprise them are the fact that you really don't own your time as a manager. Your team and your management and the company owns your time. More and more the more senior you become as a manager. I think individual contributors often think that if they become a manager, they will still have some of the freedom that they have as a senior individual contributor.

中文翻译:

是的，我想有几点。我认为……假设他们真的想做好，确实有很多人进入管理层后完全不理解这份工作，甚至没有足够的自我意识去发现自己不理解。但对于那些努力想做好的人，我认为有几件事会让他们感到惊讶：第一，作为经理，你真的不再拥有自己的时间了。你的团队、你的上级和公司拥有你的时间。你职位越高，这种情况就越严重。我认为 IC 经常认为，如果他们成为经理，他们仍然会拥有作为资深 IC 时的那种自由。

(00:37:08) Camille Fournier (Continued)

English:

But then they'll also be able to tell people what to do and they'll have all this authority. And the reality is, management is much more ... I'm not a huge fan of servant leadership exactly, but management really is a service job. You are serving the team, you are serving the company. Your job is to help make things better and that usually doesn't mean that you're making all the decisions. It usually doesn't mean that you snap your fingers and people jump. Because if you try that, especially in tech, right? People are just going to revolt. They're not going to listen to you. It's just too hard to have ... that's not the culture that we live in.

中文翻译:

但同时他们还能发号施令，拥有所有这些权威。而现实是，管理更多是……我并不完全是“仆人式领导”的忠实粉丝，但管理确实是一份服务性工作。你在为团队服务，为公司服务。你的工作是帮助事情变得更好，而这通常并不意味着由你做所有决定。通常也不意味着你打个响指，别人就得跳起来。因为如果你尝试那样做，特别是在科技界，人们会反抗的。他们不会听你的。这太难了……这不是我们所处的文化。

(00:37:53) Camille Fournier (Continued)

English:

And I don't think that's a good culture. I don't think that command and control, I tell you what to do and you do it. It just doesn't create creativity, right? It's the same thing as like PMs, trying to have all the ideas, right? No, you've got all these brilliant people working for you on a team, your job as a manager is not to tell them what to do in every single case. Occasionally, yes, a lot more. If you're trying to convince them of what you think should happen. In some ways, you're sort of nudging, you're encouraging, you're directing, you're setting guardrails for processes or behaviors or whatever, but it's not this glorious fearless leader. I make all these decisions and everyone looks up to me and it's awesome kind of job. It's much more grueling, much more ... you are really just sort of reacting to things in the moment. And it can be very ... it's a hard job. I do think particularly management when done well, when you're really trying to do it in a thoughtful kind, but also, productive way is a very hard job.

中文翻译:

我不认为那是一种好的文化。我不认为“命令与控制”——我告诉你做什么你就做什么——能激发创造力。这和 PM 想要垄断所有想法是一样的。不，你的团队里有这么多才华横溢的人，作为经理，你的工作不是在每种情况下都告诉他们该做什么。偶尔需要，但更多时候，你是试图说服他们接受你认为应该发生的事情。在某种程度上，你是在轻推、鼓励、引导，为流程或行为设置“护栏”（guardrails），但这不是那种“光荣无畏的领导者，我做所有决定，每个人都仰慕我，这工作太棒了”的样子。它更累人，更多时候你只是在对当下的事情做出反应。这是一份艰苦的工作。我确实认为，特别是当你想以一种周到、友善且高效的方式做好管理时，这是一份非常艰苦的工作。

(00:39:06) Lenny Rachitsky

English:

I wonder if engineering is where most ... where the highest percentage of people that move into management move back to IC. I've seen that a bunch and I wonder if engineers are the most common.

中文翻译:

我在想，工程领域是不是转型管理后又转回 IC 比例最高的地方。我见过很多这种情况，我在想工程师是不是最普遍的。

(00:39:16) Camille Fournier

English:

I don't know, but-

中文翻译:

我不知道，但是——

(00:39:18) Lenny Rachitsky

English:

After realizing what you just said is true, like, "Oh, what I done-"

中文翻译:

在意识到你刚才说的都是真的之后，他们会想：“噢，我都干了些什么——”

(00:39:22) Camille Fournier

English:

So do you not think product managers also do this? Because I think product managers also actually suffer from exactly this kind of-

中文翻译:

那你觉得产品经理不会这样吗？因为我觉得产品经理实际上也深受这种……困扰。

(00:39:28) Lenny Rachitsky

English:

They do.

中文翻译:

他们确实会。

(00:39:28) Camille Fournier

English:

Maybe sometimes-

中文翻译:

也许有时——

(00:39:28) Lenny Rachitsky

English:

They do.

中文翻译:

他们会的。

(00:39:28) Camille Fournier

English:

Yeah.

中文翻译:

是的。

(00:39:30) Lenny Rachitsky

English:

But interestingly, I was an engineering manager earlier in my career. I really did not like it. I was very unhappy in that role. As a PM manager though, I was very happy, it was a lot easier.

中文翻译:

但有趣的是，我职业生涯早期做过工程经理。我真的很不喜欢。在那个职位上我很不开心。但作为 PM 经理，我非常开心，而且容易得多。

(00:39:43) Camille Fournier

English:

Yeah, because PMs are way easier to manage. PMs are awesome to manage. PMs want to ... they're just so helpful. They want to do this ... they're good communicators. I love managing PMs. I have to say, I have to say, just my experience, engineers are such a pain. They're all primadonnas. I am an engineer, but PMs are more fun to manage my experience. So actually, you have a point. You have a point.

中文翻译:

是的，因为 PM 好管得多。管理 PM 太棒了。PM 想要……他们非常有帮助。他们想做这个……他们是优秀的沟通者。我喜欢管理 PM。我不得不说，根据我的经验，工程师太麻烦了。他们一个个都像“娇生惯养的明星”（primadonnas）。我自己就是工程师，但管理 PM 更有趣。所以实际上，你说得对。

(00:40:12) Lenny Rachitsky

English:

But I wonder. Yeah, because I haven't seen a lot of PM managers move back to IC product management. I find that once you can build product through teams and not sit there all day in Excel and check in on deadlines and things, it's hard to give that part up. You kind of enjoy being higher up in that chain. Yeah. Kind of along these lines, something that you have a really interesting perspective on is one-on-ones. Most people are like have one-on-ones with everyone, have them regularly, they're really important. You actually have this contrarian take that maybe you should have less one-on-ones, especially as an engineering manager, you talk about that.

中文翻译:

但我很好奇。是的，因为我没见过很多 PM 经理转回 IC 产品管理。我发现，一旦你能通过团队构建产品，而不是整天坐在那里弄 Excel、盯着截止日期之类的，就很难放弃那部分。你会享受在链条中处于更高位置的感觉。是的。顺着这个思路，你对“一对一面谈”（1:1s）有一个非常有趣的观点。大多数人都觉得应该和每个人做 1:1，定期做，这很重要。你实际上有一个相反的观点，认为也许你应该减少 1:1，特别是作为工程经理。你能谈谈吗？

(00:40:46) Camille Fournier

English:

Yeah, so to be clear, you should have one-on-ones with your direct reports and your manager and you should hold those sacred ... I tend to do my weekly or maybe every other week. So this is not about that set of one-on-ones. So the one-on-ones of the people that you are directly managing and with your own manager. But I think there is this ... I don't know if it's the remote work, sort of explosion that's happened or it's big companies or what, but what I've seen and I've heard a lot of friends at many companies complain about is this idea that everybody is doing one-on-ones with everyone else. So the manager is doing one-on-ones with their team. They're also doing one-on-ones with all of their peers. They're doing one-on-ones with all of their stakeholders. They're doing one-on-ones with product and design. And I think that it's just not a scalable approach, right? Linearly scale with the number of relationships you have. And so, as your company grows, as the number of things your team supports grow, as the team grows, you just can't scale that way. You cannot expect to maintain a one-on-one approach to kind of organizational relationship building and awareness passed a fairly small team/company.

中文翻译:

是的，首先要明确，你应该和你的直属下属以及你的经理进行 1:1，并且应该视其为神圣不可侵犯的……我倾向于每周或每两周做一次。所以这指的不是这部分 1:1。我说的是除了直属下属和经理之外的。我认为现在有一种现象……我不知道是因为远程办公的爆发，还是大公司的原因，但我看到并听到很多公司的朋友抱怨：每个人都在和所有人做 1:1。经理在和团队做 1:1，还在和所有同级做 1:1，和所有利益相关者（stakeholders）做 1:1，和产品、设计做 1:1。我认为这根本不是一种可扩展的方法，对吧？它随着你关系数量的增加而线性扩展。因此，随着公司成长、团队支持的项目增多、团队规模扩大，你根本无法以这种方式扩展。你不能指望在超过一定规模的小团队或小公司之后，还靠 1:1 来建立组织关系和保持认知。

(00:42:30) Camille Fournier (Continued)

English:

I also think that people think that one-on-ones will solve all of their problems. And I think the reality is you have this one-on-ones with people that don't really want to have a one-on-one with you. If they're not in the mind to invest in your relationship, if they don't really care about what you're doing, they actually may not like it that you're asking them for a one-on-one, let me show up like, because it's like, okay, well I should do this to be a good corporate citizen or whatever. But they're just like, why are we doing this? What is the point of this? I also think that one-on-ones, particularly when you use them first of stakeholder management. So when you're meeting with people that's not your team, but other stakeholders that you may need to deal with, if you have a lot of stakeholders ... so I've been in a lot of positions where I have a lot of internal stakeholders because I build internal platforms is a big part of what my job has been in the last many years.

中文翻译:

我还认为，人们觉得 1:1 能解决所有问题。而现实是，你在和那些并不真正想和你做 1:1 的人面谈。如果他们没打算在你们的关系上投入，如果他们并不真正关心你在做什么，他们实际上可能并不喜欢你找他们做 1:1。他们

参加只是因为：“好吧，为了当个好的企业公民，我应该参加。”但他们心里想的是：“我们为什么要干这个？这有什么意义？”我还认为，特别是当你把 1:1 用于利益相关者管理时——也就是当你见的人不是你的团队成员，而是你需要打交道的其他利益相关者。如果你有很多利益相关者……我曾担任过很多职位，有很多内部利益相关者，因为在过去的许多年里，构建内部平台是我工作的很大一部分。

(00:43:07) Camille Fournier (Continued)

English:

Having all these meetings as one-on-ones with those stakeholders can actually be kind of a weakness because when your stakeholders just tell you in a one-on-one that they're happy or unhappy with things, your unhappy stakeholders aren't hearing that. So you may have one really unhappy stakeholder and five really happy ones and all you're saying to your unhappy one is, "Well, everybody else is happy and they're not seeing it for themselves." And they're just having to say, "Trust me, because I've had these other one-on-ones with all these people and they're saying it's fine." And it just kind of sounds whiny. And so, when you're dealing with a lot of stakeholders and trying to just rely on one-on-one management of that is actually not very productive in a lot of ways. So in general, I guess I just think people should respect their own time more. As much as I just said management, you're kind of at everyone's mercy and that is true, but also respect your time. Don't just load yourself up with meetings because you're a manager and that's your job. Do you really have something to talk to a person about? Do you really need to ... when you have a one-on-one meeting with someone, you are asking for their time. You shouldn't just be doing that kind of haphazardly for fun.

中文翻译:

把所有这些与利益相关者的会议都搞成 1:1 实际上可能是一种弱点。因为当你的利益相关者在 1:1 中告诉你他们对某些事满意或不满意时，其他不满意的利益相关者听不到。所以你可能有一个非常不满意的利益相关者和五个非常满意的，而你对那个不满意的人只能说：“嗯，其他人都很满意。”但他们没亲眼看到。他们只能听你说：“相信我，因为我和其他人做过 1:1，他们都说没问题。”这听起来有点像在抱怨。所以，当你处理大量利益相关者时，仅仅依靠 1:1 管理在很多方面其实并不高效。总的来说，我认为人们应该更尊重自己的时间。虽然我刚才说管理就是听凭大家支配，这确实是真的，但也要尊重你的时间。不要仅仅因为你是经理、这是你的工作，就给自己塞满会议。你真的有事要和那个人谈吗？你真的需要……当你和某人做 1:1 时，你是在占用他们的时间。你不应该只是为了好玩而随随便便地这么做。

(00:44:24) Camille Fournier (Continued)

English:

This is a little bit of my personality. I'm not a great company networker. Some people are really good at just like, let's go to get coffee, let's go to lunch and let's hang out and build a relationship. And honestly, those people are really successful in many ways that I am not. I do sort of envy that skill. I'm really good if I work with you. If I work with you on something, generally speaking, people really come to respect me because I'm very engaged and I'm a really good collaborator in various ways, but I'm really bad at just getting to know you random one-on-one, where we don't have a purpose to the meeting. And that's not to say those are always bad, but I do think that a lot of people with ... that's your comfort zone, if that getting to know you, random, relationship building one-on-one is your comfort zone, I should probably do more of them. You should probably do fewer of them because you should always be pushing yourself to get out of your comfort zone and are you really getting something out of that or are you just being able to tick a checkbox that says, I had eight meetings today, therefore I was productive.

中文翻译:

这有点涉及我的性格。我不是一个擅长在公司搞社交的人。有些人非常擅长“咱们去喝杯咖啡吧”、“一起吃个午饭吧”，通过闲聊来建立关系。老实说，那些人在很多我不擅长的方面非常成功。我有点羡慕那种技能。如果我和你一起工作，我表现会很好。如果我和你合作某件事，通常人们会非常尊重我，因为我非常投入，而且在各方面都是很好的协作者。但我真的很不擅长那种没有目的的随机 1:1 结识。这并不是说那些会议总是不好，但我认为对于很多人来说……如果那是你的舒适区，如果那种随机的、建立关系的 1:1 是你的舒适区，那我可能应该多做一些，而你可能应该少做一些。因为你应该总是逼自己走出舒适区。你真的从中得到了什么吗？还是你只是为了在待办清单上打个勾，说“我今天开了八个会，所以我很有生产力”？

(00:45:26) Lenny Rachitsky

English:

Yeah, that super resonates. Kind of pulling on this thread of how you operate and how you work something that I've heard you're really known for, is creating a work culture where people work really hard but also, have a really good work-life balance. And when we were chatting, you, you actually told me you're not a great believer in working too hard. Can you just talk about this philosophy on building a great culture where people don't work too hard but also, get stuff done and don't burn out?

中文翻译:

是的，这非常有共鸣。顺着你运作和工作方式的思路，我听说你非常出名的一点是：创造一种工作文化，让人们既能努力工作，又能保持良好的工作与生活平衡。当我们聊天时，你实际上告诉我，你并不太相信“工作太努力”。你能谈谈这种建立优秀文化的哲学吗？即人们不超负荷工作，但依然能完成任务且不会职业倦怠？

(00:45:51) Camille Fournier

English:

I think we all understand that if we could just figure out and focus on really the most important things, we would just ... everything would be better. And it's hard to figure out what the most important things are, but overwork kind of lets you just sidestep doing the hard work of figuring out what's important in the first place. I listened to one of your podcasts where you talked to someone who said, if you've never fired someone and regretted it, you don't know where the line is, of who you should and shouldn't buy. And I will admit I have mixed feelings about that, although I get what you're saying. If you don't regularly reset your expectations of what you should and shouldn't let slide, do you have any idea where the line of what is actually important to work on is?

中文翻译:

我认为我们都明白，如果我们能弄清楚并专注于真正最重要的事情，一切都会变得更好。弄清楚什么最重要很难，而过度工作某种程度上让你回避了“弄清楚什么最重要”这项艰巨的工作。我听过你的一期播客，你和某人聊天，他说如果你从未解雇过某人并感到后悔，你就不知道该留谁不该留谁的界限在哪里。我承认我对这种说法心情复杂，尽管我明白你的意思。如果你不定期重新审视你对“什么该放手、什么不该放手”的预期，你根本不知道真正重要的工作界限在哪里。

(00:46:41) Camille Fournier (Continued)

English:

I just think ... and the thing about that is it's not like firing people where you do it once or twice and you regret it and you learn pretty quickly, unfortunately. I think you actually have to kind of regularly test the circumstances and challenge with, am I really getting the most out of myself? Am I really producing the

best value or am I just making ... swaging my internal guilt about I need to work 60-hour weeks, I need to sleep in the office, I need to whatever, to show that I'm a hard worker and I care. And so, I guess I do really think that people should challenge themselves to be focused and get the important stuff done and always be asking themselves what is important to do?

中文翻译:

我只是觉得……而且这件事不像解雇人，你做一两次，后悔了，不幸的是你很快就能学会。我认为你实际上必须定期测试环境并挑战自己：我真的发挥出最大潜力了吗？我真的创造了最大的价值吗？还是我只是在……平复内心的愧疚感，觉得我需要每周工作 60 小时，我需要睡在办公室，我需要做这个那个，来证明我是一个努力工作且在乎公司的人。所以，我确实认为人们应该挑战自己保持专注，完成重要的事情，并不断问自己：什么是重要的？

(00:47:37) Camille Fournier (Continued)

English:

And what's important to do does change over time. But if you're not regularly doing an audit of your time and trying to knock things off that list that don't matter, you're probably wasting a lot of time on things that don't matter and okay, you don't want to work less. You love working a lot, that's fine, but you could probably just be getting a lot more valuable stuff done if you would do these audits regularly, right? I also think people don't delegate enough. So, if you don't delegate, then you get overwhelmed because your team doesn't know how to do anything, because you haven't bothered to spend the time delegating, which actually takes more time initially, usually, you have to teach people whatever it is that you're trying to get them to do. Not always, sometimes they'll surprise you and they're better at it than you are, but maybe not. And so, you have to teach them, but then you finally ... you freed yourself up to scale. So I guess I'm just a real believer that working hard and a focused way for over fewer hours I think is a more productive way to approach work. And I think I have lived my career that way. I've been successful in it and I have encouraged it and people who have worked for me, and I think I've seen a lot of success through it, but it's not a thoughtless exercise. It's an active process of constant reflection to get to that focus.

中文翻译:

重要的事情会随时间而改变。但如果你不定期审计自己的时间，并尝试从清单中剔除那些不重要的事情，你可能在不重要的事情上浪费了大量时间。好吧，你不想少工作，你热爱长时间工作，这没问题，但如果你定期做这些审计，你可能会完成更多有价值的事情，对吧？我还认为人们授权（delegate）不够。如果你不授权，你就会不堪重负，因为你的团队什么都不会做，因为你没舍得花时间去授权。授权起初通常会花费更多时间，你必须教别人你想让他们做的事。不总是这样，有时他们会让你惊喜，做得比你还好，但也可能不会。所以你必须教他们，但最终……你解放了自己，实现了规模化。所以我坚信，以专注的方式在更短的时间内努力工作，是更高效的工作方式。我的职业生涯一直是这样度过的，我取得了成功，我也鼓励我的下属这样做，我看到了很多成功的案例。但这并不是一个盲目的过程，而是一个不断反思以达到专注的主动过程。

(00:49:11) Lenny Rachitsky

English:

For someone that is listening to this and is like, I want to start doing this, is there anything ... any specific practice you've found useful or any specific tactic to actually do this well?

中文翻译:

对于正在听这段话并想开始尝试的人，你有没有发现什么有用的具体做法或战术，能真正把这件事做好？

(00:49:23) Camille Fournier

English:

I think there's different approaches you could take, right? One approach you could take is just like every Friday I'm going to stop working at 4 PM or something, let's say, and I am not, or I'm not going to let myself work this weekend. Forcing yourself to log off, forcing yourself to have boundaries and saying ... and then doing out of what did I get done, I think can help. That's scary. And people don't like doing that, but I do think that that's one of the best ways to do it is really just saying, I'm going to log off. I'm going to log off every day this week at 6 PM. I'm going to ... because your brain is probably going to keep thinking. You might still be a little stressed out, you're still thinking about work, you're still worrying about it, but there is a difference between thinking about it and doing it, and particularly for those of you who are earlier in your careers, this was actually, I think one of the reasons that I did get to mastery was because I was extremely good at being focused when I was at work, when I was a hands-on programmer and really writing code for many of the hours of the day.

中文翻译:

我认为可以采取不同的方法。一种方法是：比如每个周五我下午 4 点就下班，而且我不允许自己在这个周末工作。强迫自己下线，强迫自己设定界限，然后审视“我完成了什么”，我认为这会有帮助。这很吓人，人们不喜欢这样做，但我认为这是最好的方法之一，就是直接说：“我要下线了。这周我每天下午 6 点准时下班。”因为你的大脑可能会继续思考，你可能仍然有点压力，仍在想工作，仍在担心，但“思考”和“动手做”是有区别的。特别是对于那些处于职业生涯早期的人，我认为我之所以能达到精通，原因之一就是我在工作时极其专注。当我是一名一线程序员时，我每天有很多小时都在真正地写代码。

(00:50:10) Camille Fournier (Continued)

English:

And that meant I was not ... this was pre heavy social media. I was much less distracted, which I don't know if I would be able to do it in the modern era as easily, but having that real heavy focus time because I was like, "I don't want to work on the weekend. I don't want to stay here until 9 PM every night. I just want to get this done." And it meant that I didn't do quite as many copies. I didn't go to lunch and chat with people all every day, but I was very, very productive and I learned to get a lot done in a short period of time. And I do think learning those skills earlier in your career and then continuing to apply them throughout your career is another piece of advice.

中文翻译:

那意味着我没有……那是社交媒体还没那么泛滥的时代。我受到的干扰少得多，我不知道在现代我是否还能那么容易做到。但我有那种高强度的专注时间，因为我想：“我不想周末加班。我不想每晚待到 9 点。我只想把这事干完。”这意味着我没喝那么多咖啡，没每天都去吃午饭和人闲聊，但我非常非常高效，我学会了在短时间内完成大量工作。我认为在职业生涯早期学习这些技能，并在整个职业生涯中继续应用它们，是另一个建议。

(00:51:16) Lenny Rachitsky

English:

In terms of staying and getting focused, is there anything that helps you focus? Is it like headphones, a drink? What gets you in the zone?

中文翻译:

关于保持和进入专注状态，有什么能帮到你吗？是耳机、饮料？什么能让你进入状态（in the zone）？

(00:51:24) Camille Fournier

English:

I mean, yeah, I have a lot of rituals. I have my caffeination rituals.

中文翻译:

是的，我有很多仪式。我有我的咖啡因仪式。

(00:51:31) Lenny Rachitsky

English:

Say more.

中文翻译:

展开说说。

(00:51:32) Camille Fournier

English:

Well, I mean I can't drink coffee anymore because my stomach got messed up at some point, but I drink tea in the morning. I drink caffeinated water also. I have a Diet Coke at lunch, so I have these rituals of my caffeine hits that help me. I have to be in a quiet place. I do find non-music that is ... I really like Four Tet, for example, as music to focus or the new Andre 3000 Flute album is extremely good for focusing, where it's not quite predictable, no lyrics and not quite predictable. For me, that helps me focus a lot. I can't be listening to any words and focus. My brain just cannot ignore words. So those are some of the things I use to help folks.

中文翻译:

嗯，我不能再喝咖啡了，因为某个时候我的胃搞坏了。但我早上喝茶，还喝含咖啡因的水。午餐时喝健怡可乐。我有这些咖啡因摄入仪式来帮我。我必须待在安静的地方。我发现一些非音乐类的……比如我非常喜欢 Four Tet 的音乐来专注，或者 Andre 3000 的新长笛专辑也非常适合专注，因为它们不太可预测，没有歌词。对我来说，这非常有助专注。我不能听任何有歌词的东西，我的大脑无法忽略文字。这些就是我用来帮助专注的方法。

(00:52:24) Lenny Rachitsky

English:

That is awesome. I have a similar caffeine strategy. I'm drinking tea always during these podcasts and this little thing I got here and then, my wife doesn't want me to drink Diet Coke because she thinks the sugar and it is not good, but I still drink it sometimes and it works great. I switched to green tea. That's my approach. Start with black tea and then switch to green tea.

中文翻译:

太棒了。我有类似的咖啡因策略。录播客时我总是在喝茶。我妻子不想让我喝健怡可乐，因为她觉得里面的代糖不好，但我有时还是会喝，效果很好。我改喝绿茶了。这是我的方法：先喝红茶，再换绿茶。

(00:52:40) Camille Fournier

English:

That's smart.

中文翻译:

很聪明。

(00:52:41) Lenny Rachitsky

English:

Man, there's so many things you shared that I wanted to dig into. Let me share a couple of things. Your point about delegating I thought was really important, and there's another benefit to learning to delegate, which is team members feel empowered. You give them a chance to take on responsibility and they feel like I have an opportunity to show I can do this other thing.

中文翻译:

天哪，你分享了这么多我想深入探讨的东西。我分享几点：你关于授权的观点我认为非常重要。学会授权还有另一个好处，就是团队成员会感到被赋能。你给了他们承担责任的机会，他们会觉得：“我有机会展示我也能做别的事情。”

(00:53:02) Camille Fournier

English:

Yeah, and you're never going to scale if you don't delegate.

中文翻译:

是的，如果你不授权，你永远无法实现规模化。

(00:53:06) Lenny Rachitsky

English:

Yeah, and it's hard to start learning to do that, but once you get good at it, it becomes so great. I love just delegate everything every time, and people enjoy it. They're like, "Amazing. You're giving me opportunity." The thing you said about how ... if you're not sometimes regretting something you cut or potentially firing someone you don't regret, Elon actually, Elon Musk has a great approach to this. I don't know if you've heard his philosophy on optimizing. He has this whole five-step process of figuring out how to optimize something, and one of them is try to cut things like, do we need this thing or not? And if you don't recognize that 10% of stuff you cut, it was a mistake, you're probably not cutting enough stuff.

中文翻译:

是的，刚开始学很难，但一旦你擅长了，感觉就太棒了。我喜欢每次都把所有事情授权出去，大家也很享受，他们会说：“太棒了，你给了我机会。”你提到的关于……如果你不偶尔后悔砍掉了某些东西，或者解雇了某

人。埃隆・马斯克（Elon Musk）对此有一个很好的方法。我不知道你是否听过他的优化哲学。他有一套完整的五步流程来优化事物，其中一步就是尝试砍掉东西，比如：“我们到底需不需要这个？”如果你没发现你砍掉的东西里有 10% 是个错误，那你砍得可能还不够多。

(00:53:45) Camille Fournier

English:

You'll have to figure out your own metrics. But I definitely think testing the limits, it's scary though. I'm going to be honest. Doing that is always scary. It brings up like, I forgot to finish the assignment nightmares of school, especially for the overachievers that often end up in these kinds of companies and jobs. Again, you got to do things that scare you or you're never going to grow.

中文翻译:

你得找到自己的衡量标准。但我绝对认为测试极限很重要，虽然这很吓人。说实话，这样做总是很吓人。它会勾起那种“忘了写作业”的学校噩梦，特别是对于那些经常进入这类公司和职位的“优等生”来说。再说一次，你必须做那些让你害怕的事，否则你永远不会成长。

(00:54:14) Lenny Rachitsky

English:

Okay, I'm going to go in a whole different direction. You've got a book coming out about platform engineering and platform teams. This is something that I get a lot of questions about. A lot of people are thinking about moving to a platform team or struggling working on a platform team or struggling working with a platform team. So I have a bunch of questions along these lines. The first is I asked a bunch of people that worked with you what to ask you and someone that worked with you shared this quote about his frustration working with platform teams that he's often complaining to you about and he works on customer-facing products. And so, he said platform teams that are often slow. They're often pushing us to compromise on features to adopt their systems. They often get infinite funding even though they don't have to show any ROI. And so, maybe from the perspective of working with a platform team, say you're building something customer-facing, any tips for effectively working with a platform team and building a great relationship there.

中文翻译:

好，我要换个完全不同的方向。你有一本关于平台工程和平台团队的新书即将出版。这是我收到很多提问的话题。很多人在考虑转到平台团队，或者在平台团队工作得很挣扎，或者与平台团队合作得很挣扎。所以我有一系列相关问题。首先，我问了一些曾与你共事的人该问你什么，其中一位分享了他与平台团队合作时的挫败感（他经常向你抱怨），他是做面向客户的产品的。他说平台团队通常很慢，经常逼我们在功能上妥协以采用他们的系统，而且他们经常获得无限的资金，即使他们不需要展示任何投资回报率（ROI）。所以，从与平台团队合作的角度来看（假设你在构建面向客户的东西），有没有什么建议可以有效地与平台团队合作并建立良好的关系？

(00:55:08) Camille Fournier

English:

I'm very sympathetic to anybody who feels that way because part of the reason that I wrote this book, I co-wrote it with a friend, Ian Nolan, is that so many platform teams are guilty of all the things that he's complaining about, that my friend was complaining about, that they don't listen, they're not delivering

effectively, they're not explaining their value to the company. And it is infuriating when you're dealing with that. And honestly, I'm very sympathetic. The thing that I would say though is the more you can, first of all, spend the time understanding that the platform team's problems a little bit and trying to collaborate and work with them and be clear about what it is you actually need and how you're willing to work with them, I think the better.

中文翻译:

我非常同情有这种感觉的人，因为我写这本书（我和朋友 Ian Nolan 合著）的部分原因就是，太多的平台团队确实犯了那位朋友所抱怨的所有错误：不听取意见、交付效率低下、不向公司解释自己的价值。遇到这种情况确实让人抓狂。老实说，我很同情。但我会说，首先，你越能花时间去了解平台团队面临的问题，尝试与他们协作，并明确表达你真正需要什么以及你愿意如何与他们合作，效果就会越好。

(00:56:01) Camille Fournier (Continued)

English:

I think if you get mad and you just try to avoid them or undermine them or whatever, that may work in the long run, but it's just going to make everything worse in the short run. So I do think that if you've got a platform team that you're frustrated with, some tips, I would put are like, "Look, find the parts of that team that are good because I'm sure there are some," right? Maybe the databases team is awesome. And really make sure you are maintaining a good relationship with that part of the team and you can point to how you are collaborating extremely well with that part of the team. Help give them product feedback. This is annoying but sometimes needs to happen because a lot of companies, I actually think product ... you need to have a product mindset and frankly, you need to have product managers to build good platforms, internal platforms.

中文翻译:

我认为如果你只是生气，然后试图避开他们或暗中破坏，从长远来看可能有用，但短期内只会让事情变得更糟。所以如果你对某个平台团队感到沮丧，我的建议是：“寻找那个团队中表现好的部分，因为我相信肯定有”，对吧？也许数据库团队就很棒。确保你与那部分团队保持良好关系，并能展示你是如何与他们高效协作的。帮助他们提供产品反馈。这很烦人，但有时必须这样做，因为在很多公司，我认为你需要有产品思维，坦白说，你需要有产品经理来构建好的平台（内部平台）。

(00:57:16) Camille Fournier (Continued)

English:

A lot of companies just don't do this. They don't believe that they should waste head count on product managers for internal teams. So you end up with this platform team that has a lot of smart engineers, but they don't really know what to build. So they're just sort of building whatever they think is right. And so sometimes your job is to product manage them, is to tell them, these are the problems that we have and this is what we need. And the clearer you can be about that, particularly when they don't have a product team, oftentimes, you can kind of lead them from the side in that way. And so I think that's another approach that I would take when you're in that situation. Find the parts of the team that are working and working well with your team and make sure you develop those relationships and try to just get over the anger and frustration and just be clear about what it is that you need and help them understand what they should really be doing and building.

中文翻译:

很多公司不这样做。他们不认为应该把人力名额浪费在内部团队的产品经理身上。结果就是，你的平台团队有很多聪明的工程师，但他们并不知道该造什么。所以他们就随心所欲地造一些他们认为正确的东西。所以有时你的工作就是去“产品管理”他们，告诉他们：“这是我们面临的问题，这是我们需要的东西。”你表达得越清晰，特别是当他们没有产品团队时，你往往可以从侧面引导他们。所以这是我在那种情况下会采取的另一种方法。找到团队中运作良好且与你团队配合默契的部分，建立这些关系，尝试克服愤怒和挫败感，明确你的需求，并帮助他们理解他们真正应该做和构建的是什么。

(00:57:48) Lenny Rachitsky

English:

That is really interesting advice. So especially, you're saying if they don't have a PM helping, make sure they're guiding things well, is you can help them ... basically help them think through stuff that will benefit them and also, help the stuff that you're trying to get done.

中文翻译:

这建议非常有意思。所以特别是当他们没有 PM 协助引导时，你可以帮助他们……基本上是帮他们思考那些既对他们有利，又能帮你完成任务的事情。

(00:58:03) Camille Fournier

English:

Yeah.

中文翻译:

是的。

(00:58:04) Lenny Rachitsky

English:

That's awesome advice. Okay, so what about from the leadership perspective, trying to build an effective platform team, do you have any advice on how to structure these teams and incentivize these teams to help them be successful?

中文翻译:

很棒的建议。那么从领导层的角度来看，如果想建立一个高效的平台团队，你对于如何构建这些团队的结构以及如何激励他们取得成功有什么建议吗?

(00:58:17) Camille Fournier

English:

First of all, I'm a very strong believer that platform engineering has to involve software engineering. If you don't have any software engineers on your platform team and you only have more operations systems engineers, DevOps, SRE, which I realize there are some SREs that are software engineers, but they tend to not want to write big software. I think you're kind of missing the picture. Platform engineering is not just maintaining cloud infrastructure and doing small scripts or blueprints or enablement projects for other teams, because that doesn't really create a cohesive and coherent platform. It doesn't create products

and frankly you need to be ... platforms are products, ultimately. You should be thinking about how do I create coherent offerings that make this company more productive?

中文翻译:

首先，我坚信平台工程必须包含软件工程。如果你的平台团队里没有软件工程师，只有运维系统工程师、DevOps、SRE（我知道有些 SRE 也是软件工程师，但他们往往不想写大型软件），我认为你就没看清全貌。平台工程不仅仅是维护云基础设施、写点小脚本或蓝图，或者为其他团队做赋能项目，因为那并不能真正创建一个有凝聚力、连贯的平台。那不是在创造产品，而坦白说，平台最终就是产品。你应该思考的是：我如何创建连贯的产品服务，让公司更高效？

(00:59:04) Camille Fournier (Continued)

English:

So you need software engineers. Yes, you need sort of operations systems, SRE specialists as well. And of course, you need product people. I had product teams on ... product managers for all of my platform teams. I've really developed out that practice in the companies that I've worked for doing this because I just believe that you're not going to get great results if you just believe that to engineers and engineering management. They will do their best and you do occasionally find engineers that have really great product instincts in this space, but the actual details. And focus of the product work is just you can't write a bunch of code and/or manage a big software engineering team and be a product manager at the same time. That's actually just asking, I think, too much of people to do that really well, at least for very long.

中文翻译:

所以你需要软件工程师。是的，你也需要运维系统、SRE 专家。当然，你还需要产品人员。在我负责的所有平台团队中，我都有产品团队和产品经理。我在工作过的公司里一直在推广这种做法，因为我坚信，如果你只依靠工程师和工程管理，是得不到好结果的。他们会尽力而为，你偶尔也会发现一些在这个领域有极佳产品直觉的工程师，但产品工作的实际细节和专注度是不同的。你不可能在写大量代码或管理大型软件工程团队的同时，还兼任产品经理。我认为要求一个人把这两件事都做好（至少是长期做好）实在是要求太高了。

(00:59:49) Camille Fournier (Continued)

English:

So I think when it comes to structuring a team, recognize this is not just like SRE V2. This is more than that. You want software engineers, you want systems engineers, you want product people. And then, one of the reasons you want product folks is because you want to be thinking about impact-based, outcome-based approaches, not just like, "Well, we built this thing that seemed cool, we adopted this technology from this company because that's what everybody on the internet is talking about." Whether it's VC funded or big company. We actually thought about what are the problems the engineers at my company are facing? How can we improve their productivity or deliver leverage to this business through whatever it is we're building, right? Are we reducing the cycle time for engineering tasks? Are we solving problems that are preventing products from launching and scaling? Are we making really meaningful cost reductions or efficiencies and, in our products or in our platforms? These are some examples of measurable contributions, but I think one of the challenges is that people create these platform teams and think that they can be sort of divorced from having an impact focus because it's like, "Well, you just got to run the infrastructure and make sure it works."

中文翻译:

所以我觉得在组建团队时，要意识到这不仅仅是“SRE 2.0”。它远不止于此。你需要软件工程师、系统工程师和产品人员。你需要产品人员的原因之一是，你希望思考基于影响（impact-based）和基于结果（outcome-based）的方法，而不仅仅是：“嗯，我们造了这个看起来很酷的东西，我们采用了这家公司的技术，因为互联网上大家都在谈论它。”不管是风投支持的还是大公司的。我们实际上思考的是：我公司的工程师面临什么问题？我们如何通过所构建的东西来提高他们的生产力，或为业务提供杠杆（leverage）？我们是否缩短了工程任务的周期？我们是否解决了阻碍产品发布和扩展的问题？我们是否在产品或平台中实现了真正有意义的成本降低或效率提升？这些都是可衡量的贡献例子。但挑战之一是，人们创建了这些平台团队，却认为他们可以脱离对“影响”的关注，因为觉得“你只需要运行基础设施并确保它正常工作就行了”。

(01:01:29) Camille Fournier (Continued)

English:

And it's like, "No, you actually do still have to do ... if you do that OKR stuff or goal setting," or whatever, you've got to take a lot of the best practices of product. It's a little different in the internal world, but it's still very important if you want to do platforms.

中文翻译:

但事实是：“不，你仍然需要做……如果你做 OKR 或目标设定之类的”，你必须借鉴很多产品的最佳实践。在内部工具领域这可能有点不同，但如果你想做平台，这仍然非常重要。

(01:01:45) Lenny Rachitsky

English:

On the line of having PMs within the platform teams. Some of the best PMs I've ever worked with where PMs on platform teams, and that just tells me, it's not like where you put PMs that are just meant like that's ... where you could have some of the highest leverage because platform teams enable the rest of the company to move faster.

中文翻译:

关于在平台团队中配备 PM，我合作过的一些最优秀的 PM 就是平台团队的 PM。这告诉我，平台团队并不是随便塞个 PM 的地方，而是你可以获得最高杠杆的地方，因为平台团队能让公司的其他部门跑得更快。

(01:02:01) Camille Fournier

English:

Yeah.

中文翻译:

是的。

(01:02:02) Lenny Rachitsky

English:

And one difference there, I'm curious if you agree, is your ratio of PM to engineer can be a lot higher. You can have a lot more engineers per PM on platform teams.

中文翻译:

还有一个区别，我想知道你是否同意：PM 与工程师的比例可以高得多。在平台团队中，每个 PM 可以对应更多的工程师。

(01:02:10) Camille Fournier

English:

Yeah. Yeah, I think that's right because I think so much of the work in a platform team is not exactly product work. A lot of it is like scaling or really deep kind of technical, like I got to figure out how to actually do this technical thing or I've got to do this performance efficiency and you don't always need a product spec for that, right? It is often just a very engineering heavy task. So yes, I think the ratios do tend to be a bit different.

中文翻译:

是的。我认为是对的，因为平台团队的很多工作并不完全是产品工作。很多是关于扩展，或者是很深的技术问题，比如“我得弄清楚如何实现这个技术点”或者“我得做性能优化”，这些并不总是需要产品说明书（spec），对吧？这通常只是非常繁重的工程任务。所以是的，我认为比例确实会有所不同。

(01:02:42) Lenny Rachitsky

English:

We dove over right into this topic. Maybe it might be helpful to help people understand what is a platform team, just what are examples of teams that would be considered platform teams.

中文翻译:

我们直接切入了话题。也许帮助大家理解什么是平台团队会有所帮助，比如哪些团队会被认为是平台团队？

(01:02:51) Camille Fournier

English:

I mean, I guess the high level definition that I have of platform engineering is if you are developing and operating platforms that ... where they're trying to manage overall system complexity and deliver leverage to your business. So a lot of the teams that people think about nowadays and when they're talking about platform teams, they're talking a lot about teams that may have formerly been called dev tools. For example, so your CI/CD tooling for the company, a lot of the cloud infrastructure provisioning and tooling. If you're at a company with certain technical problems, you may very well be building semi bespoke storage systems. For example, maybe part of your platform teams. I actually think that there are ... And then, actually web frameworks ... Web mobile framework and support for that. That set of engineering can also be part of a platform offering.

中文翻译:

我对平台工程的高层定义是：如果你正在开发和运营旨在管理整体系统复杂性并为业务提供杠杆的平台。现在人们谈论平台团队时，很多是指以前被称为“开发工具”（dev tools）的团队。例如，公司的 CI/CD 工具链，大量的云基础设施配置和工具。如果你在一家有特定技术问题的公司，你很可能在构建半定制的存储系统，那可能也是平台团队的一部分。实际上我认为……还有 Web 框架、移动端框架及其支持，这套工程也可以是平台产品的一部分。

(01:04:33) Camille Fournier (Continued)

English:

I actually kind of believe that there's also sort of what you might call integration platforms or platforms that are in between that infrastructurey developer tooly stuff and products. So billing platforms for example, if you have a single billing platform for all of the different products in your company, that shares a lot of overlaps in the challenges of those more dev tools, infrastructurey platform teams because you're probably supporting lots of different product lines that are using this system that needs to be able to scale with certain efficiencies. You need to still do the product discovery. You probably have a little bit more of a business product focus than the pure internal tools teams, but that gets to the blended area.

中文翻译:

我还认为存在一种所谓的“集成平台”，或者是介于基础设施/开发工具与最终产品之间的平台。比如计费平台（billing platforms），如果你为公司所有不同的产品提供统一的计费平台，它在挑战上与开发工具或基础设施平台团队有很多重叠，因为你可能要支持许多不同的产品线，这些产品线都在使用这个需要高效扩展的系统。你仍然需要进行产品调研（product discovery）。你可能比纯粹的内部工具团队更关注业务产品，但这属于交叉领域。

(01:04:44) Lenny Rachitsky

English:

For founders that are earlier stage or companies that are smaller hearing this and they're like, "Oh, shit do I need a platform team?" So yeah, you're shaking your head, if people aren't watching on YouTube. What's a sign that's time maybe to start creating a platform team and setting aside resources for something like that?

中文翻译:

对于那些处于早期阶段的创始人或规模较小的公司，听到这些可能会想：“噢，该死，我需要一个平台团队吗？”我看你在摇头（如果大家没在看 YouTube 的话）。那么，什么时候是开始创建平台团队并为此预留资源的信号？

(01:05:00) Camille Fournier

English:

So first of all, I think it gets to be like, you have 50 plus engineers. I don't think this is the kind of thing that you start when you are 10 engineers, right? But when you are ... so there's a lot of ad hoc coordination that's probably happening amongst your engineering groups right now, where maybe you just have one ... you have your GitHub and somebody is making sure that all of that stuff is sort of working. You have a few cloud databases and you got a couple of people on each team and they kind of share notes to figure out what's going on or whatever. Okay, it's all good, but at some point, you hit either a lot of inefficiency where you're seeing the same people across a bunch of different teams. Each team has the same kind of people having to solve the same kinds of problems. It just seems like why do I have three people in every team, dealing with this instead of centralizing it and making it a little more efficient.

中文翻译:

首先，我认为通常要等到你有 50 名以上的工程师。我不认为这是你在只有 10 名工程师时该开始做的事。但在……目前你的工程小组之间可能有很多临时的协调，比如你有一个 GitHub，有人负责确保一切正常运行。你有几个云数据库，每个团队有几个人，他们互相交流一下情况。这都没问题。但在某个时刻，你会遇到效率低

下的问题：你发现不同团队的人都在解决同样的问题。每个团队都有同样类型的人在处理同样的麻烦。这看起来就像：为什么我每个团队都要有三个人处理这个，而不是把它集中起来，让它更高效一点？

(01:06:36) Camille Fournier (Continued)

English:

It also could be that you hit some core scaling issue that starts to really need a dedicated team to solve. Again, that could be developer productivity. Every development team is trying to do it their own way and everybody is just super slow because you just can't get code released and it all has to be coordinated across every different team and it's just like, what is going on? We need to fix that or you actually have a technical challenging area that needs a focused team to fix the scaling. Those are some of the signs that you may want to start thinking about a platform team, but it's really like ... generally speaking, I would not jump into it early. This is something for companies that have matured where it's worth investing and making people internally more productive and centralizing certain functions just from a cost and efficiency basis at minimum.

中文翻译:

也可能是你遇到了核心的扩展问题，真的需要一个专门的团队来解决。同样，这可能是开发者生产力问题：每个开发团队都各行其是，大家都很慢，因为代码发布不了，所有事情都得跨团队协调，简直一团糟。我们需要解决这个问题。或者你确实有一个技术挑战领域，需要一个专注的团队来解决扩展性。这些都是你可能需要考虑建立平台团队的信号。但总的来说，我不会过早介入。这是为那些已经成熟的公司准备的，在这些公司，为了提高内部生产力、从成本和效率角度集中某些功能是值得投资的。

(01:07:02) Lenny Rachitsky

English:

Say you're on a platform team, say you're an engineer or even a PM, any advice for how to be successful and thrive within that environment and be a great platform team.

中文翻译:

假设你在平台团队中，无论你是工程师还是 PM，对于如何在那种环境下取得成功、蓬勃发展并成为一个优秀的平台团队，你有什么建议吗？

(01:07:11) Camille Fournier

English:

If you want to do platforms. If you're an engineer, certainly if you're an engineer or an engineering manager, you've got to remember that half of the work is actually the operational quality, operational excellence of these things. Platform engineering is not just writing code and then throwing it over the walls to someone. Being interested and passionate about the operational challenges and scaling bits of systems I do think is fairly important, if you want to be a great platform engineer from a software engineer perspective, if you're more like a product manager, "Look, you've got to really care about ..." I think you got to be really both interested in working with really smart engineers who are going to know way more than you do about things.

中文翻译:

如果你想做平台。如果你是工程师，特别是如果你是工程师或工程经理，你必须记住，一半的工作实际上是这些东西的运维质量和运维卓越性（operational excellence）。平台工程不仅仅是写完代码然后把它“扔过墙”给别人。如果你想从软件工程师的角度成为一名优秀的平台工程师，对运维挑战和系统扩展感兴趣并充满激情是非常重要的。如果你更像是一个产品经理，听着，你必须真正关心……我认为你必须既有兴趣与那些在某些方面比你懂得多得多的聪明工程师合作。

(01:08:06) Camille Fournier (Continued)

English:

And almost being a really good, not necessarily zero to one but past one person because actually a lot of the best platform type offerings in companies start in individual application teams. An application team has a problem and they solve it for themselves and it turns out that that's actually a really good idea for how to solve that problem. So a good platform team is often looking around for those and then sort of taking them and then assimilating them and making them available to more and more people. And so, I think if you want to be in that kind of zero to one building the brand new stuff all the time, you may not be as happy in a platform team. But if you're really interested a little bit more of taking things over, stabilizing, scaling them, making them efficient, making them evolve as a company evolves, I think you'll be happier in that circumstance.

中文翻译:

而且你几乎要成为一个擅长“1 之后”的人，而不一定是“0 到 1”的人。因为实际上，公司里很多最好的平台类产品都起源于具体的应用团队。一个应用团队遇到了问题，他们自己解决了，结果证明那是一个非常好的解决方案。所以一个好的平台团队经常在寻找这些方案，然后接手它们，将它们同化，并让更多的人可以使用。所以，如果你想一直处于那种“0 到 1”、不断构建全新事物的状态，你在平台团队可能不会那么开心。但如果你对接收事物、使其稳定、扩展、高效，并随着公司的发展而演进感兴趣，我认为你在这种环境下会更快乐。

(01:09:01) Lenny Rachitsky

English:

Awesome. Is there anything else along these lines before we wrap up and yet a very exciting lightning round that you think might be helpful for folks working with platform teams or working on a platform team, building platform teams?

中文翻译:

太棒了。在我们结束并进入令人兴奋的闪电轮（lightning round）之前，关于与平台团队合作、在平台团队工作或构建平台团队，还有什么你认为对大家有帮助的吗？

(01:09:14) Camille Fournier

English:

I think there are two things that we haven't touched on or maybe three that are really hard about platform teams and that I think don't get much press, which is running platform projects is a little bit different than running agile projects. You can take a lot of the best practices you may have learned from agile type product delivery, but you are running longer, running larger scale, more complex builds because a lot of the stuff that you build at the platform level just takes longer. It's a little bit more complicated. So you've got to be willing to get into that kind of stuff, especially if you're in a management job in that space.

中文翻译:

我认为有两三件事我们还没触及，它们是平台团队真正困难的地方，而且我觉得媒体报道不多。第一，运行平台项目与运行敏捷（Agile）项目有点不同。你可以借鉴很多从敏捷产品交付中学到的最佳实践，但你运行的是周期更长、规模更大、更复杂的构建，因为平台层面的很多东西就是需要更长时间，也更复杂。所以你必须愿意投入到这类事情中，特别是如果你在那个领域担任管理工作。

(01:09:55) Camille Fournier (Continued)

English:

You are going to deal with a lot of migrations as well. So it is very annoying. Migrations happen, even if you don't force them on people, your a cloud provider is going to say, oh, you've got to migrate from EKS view 19 to 121 or whatever, and that may require changes in code and then that's a real pain in the butt for all of the application teams. So people who are interested in how to smooth over the customer and company impact of this underlying change, I think we'll be very happy in platform teams. If you're not interested in that, you may not enjoy the work as much. There's just a lot of detail oriented work and platforms.

中文翻译:

你还要处理大量的迁移工作。这非常烦人。迁移总会发生，即使你没强迫别人，你的云服务商也会说：“噢，你得从 EKS 1.19 迁移到 1.21”之类的，这可能需要更改代码，对所有应用团队来说都是件麻烦事。所以，那些对如何平滑底层变更对客户和公司的影响感兴趣的人，在平台团队会很开心。如果你对此不感兴趣，你可能不会那么享受这份工作。平台工作中有很多注重细节的任务。

(01:10:55) Camille Fournier (Continued)

English:

And then of course, the final part is the stakeholders are a nightmare. My friend who wrote the question about how his product or his platform partners drive him crazy. I have no doubt he drives them crazies. Because you've got all of these stakeholders, your peers and tech, maybe product managers, maybe executives that are like, what is this team? Are they really worth the money? Why are we spending so much on it? I don't understand what they're doing. I could do it better in some cases. So there's actually a big part of the job, particularly as either product managers or engineering leadership is stakeholder management and really learning how to do that well. And that is not always, that's probably I think universally agreed to be the least fun part of the job. I don't think anybody loves it, but it is certainly a skill set. I'm glad that I have. And so, I think if you are thinking about building one of these teams. And you're like, I can just do the fun tech stuff or the cool product stuff, I'm really passionate about engineering productivity or I'm really passionate about state scale storage systems and I don't want to think about any of the other stuff. You may not actually be happy in the long run in leadership positions. It may be fine as an individual contributor, but for leaders in particular, there's a lot more to it than the fun tech problems, the fun operations problems, even the product.

中文翻译:

最后一部分当然是：利益相关者简直是噩梦。那位问“平台合作伙伴如何让他抓狂”的朋友，我毫不怀疑他也让他们抓狂。因为你有这么多利益相关者：技术同僚、产品经理、高管，他们会问：“这个团队是干嘛的？他们真的值这么多钱吗？为什么我们要花这么多钱？我不明白他们在做什么。在某些情况下我能做得更好。”所以，这份工作的很大一部分（特别是作为 PM 或工程领导者）是利益相关者管理，并真正学会如何做好它。这通常被公认为工作中参与感最低的部分。我不认为有人喜欢它，但这确实是一项技能，我很庆幸我拥有它。所

以，如果你在考虑组建这样一个团队，并且觉得“我只想做有趣的技术活或酷炫的产品，我只对工程生产力或大规模存储系统感兴趣，不想考虑别的”，那么从长远来看，你在领导岗位上可能不会开心。做 IC 可能没问题，但对于领导者来说，除了有趣的技术和运维问题，甚至除了产品本身，还有很多其他事情要处理。

(01:12:07) Lenny Rachitsky

English:

You said there's a few things we haven't touched on. Is there anything else?

中文翻译:

你说有几件事我们还没触及。还有别的吗?

(01:12:13) Camille Fournier

English:

That's the fastest and anybody who's interested, my book will be coming out in the next couple of months from O'Reilly and covers all of this depth.

中文翻译:

那是概括版。任何感兴趣的人，我的书将在未来几个月由 O'Reilly 出版，涵盖了所有这些深度内容。

(01:12:22) Lenny Rachitsky

English:

Yeah, is there a date specifically people should watch out for?

中文翻译:

是的，有没有具体的日期大家需要留意的?

(01:12:25) Camille Fournier

English:

I think it's actually pre-order on Amazon now. It's called Platform Engineering, and then, there's ... I think it's like a guide for technical product and people leaders, something like that. But I think the release ... I don't know exactly when the Kindle release will be, but we're still in copy edits, but it should be done pretty soon.

中文翻译:

我想现在已经在亚马逊上预售了。书名是《平台工程》（Platform Engineering），副标题大概是“技术产品与团队领导者指南”之类的。至于发布日期……我不太确定 Kindle 版什么时候出，我们还在进行文字校对，但应该很快就会完成。

(01:12:42) Lenny Rachitsky

English:

Okay, amazing. So you can pre-order it now. Rolling to it obviously in the show notes and description, it's called Platform Engineering. Before we get to a very exciting lightning round, I want to take us to a recurring segment on this podcast that I call AI Corner. AI is very top of mind for a lot of people and I'm always curious how people are finding it valuable in their work or in their life. So the question is there anything you've found AI to be helpful with ... in your AI work? Any way you use it in an interesting way?

中文翻译:

好的，太棒了。所以现在就可以预订了。我们会在节目笔记和描述中附上链接，书名是《平台工程》。在进入令人兴奋的闪电轮之前，我想带大家进入本播客的一个固定环节，我称之为“AI 角落”。AI 是很多人关注的焦点，我总是很好奇人们如何发现它在工作或生活中的价值。所以问题是：你发现 AI 在你的工作中有什么帮助吗？有没有什么有趣的用法？

(01:13:11) Camille Fournier

English:

I find it helpful, for example, if I've written a sentence and I'm like, I like this sentence, but I feel like, I don't like the phrasing or the format that I've used, it needs to be edited, but I can't quite figure out how. I will often put it into ChatGPT and be like, "Can you reframe this? Can you rephrase this for me?" It doesn't always work well, but sometimes it does. It's like, "Oh yeah, if I just switched this around or changed this word choice, it's a much easier to read sentence." I do think it's ... I think it's just before that and small. I think it's large, lots of texts I haven't had as much luck with. I'm a little bit of an AI novice still, I would say.

中文翻译:

我发现它很有用，比如当我写了一个句子，我觉得“我喜欢这个意思，但不喜欢这个措辞或格式”，需要修改但我想不出怎么改。我经常把它扔进 ChatGPT，说：“你能帮我重构一下吗？帮我换个说法。”它并不总是奏效，但有时确实有用。我会想：“噢对，如果我把这个调换一下或者改个词，句子读起来就顺畅多了。”我觉得它在处理这类小细节上很棒。处理大段文本我还没那么好运。我想说，我仍然是个 AI 新手。

(01:13:52) Camille Fournier (Continued)

English:

What I will say is that AI is really bad, if you try to ask it for quotes or at least ChatGPT is. I haven't used a lot of the other ones that much. I actually saw there was some Twitter or some social media thing that's like, did Francis Ford Coppola get fake reviews from ChatGPT of the new, what is it, Agopolus.

中文翻译:

我想说的是，如果你向 AI 索要名言，它表现得很差，至少 ChatGPT 是这样。我没怎么用过其他的。我实际上看到 Twitter 或社交媒体上说，弗朗西斯·福特·科波拉（Francis Ford Coppola）是不是从 ChatGPT 那里得到了关于他新片（叫什么来着，《大都会》）的虚假评论。

(01:14:14) Lenny Rachitsky

English:

Yeah.

中文翻译:

是的。

(01:14:14) Camille Fournier

English:

Agopolus maybe.

中文翻译:

可能是《大都会》（Megalopolis）。

(01:14:14) Lenny Rachitsky

English:

Yeah.

中文翻译:

是的。

(01:14:16) Camille Fournier

English:

And I was like, I actually had that scenario where I was like, I need a quote ... I was looking for quotes for the book and I was like, maybe ChatGPT knows, but I was like, "Can you give me any good quotes on ..." I forget what it was. Let's just say it was on managing complexity or something like that, and it gave me these really interesting quotes. I was like, that's great, but I better double check that. And they were not real. They were never real, not a single quote. And I'd be like, that's not actually a real quote. Yes, you're right. That's not, so here's a different one. I was like, that's still not a real quote, so don't ask it for quotes because ... or if you do, make sure it's a real quote and not just an interestingly phrased ... I mean good summarization in some ways of the text, it was sort of quoting from, just not an actual quote.

中文翻译:

我当时想，我确实遇到过那种情况：我需要一句名言……我在为书寻找引用，我想也许 ChatGPT 知道。我问：“你能给我一些关于……（我忘了是什么了，假设是关于管理复杂性）的精彩名言吗？”它给了我一些非常有趣的引用。我想：“太棒了，但我最好核实一下。”结果它们全是假的。没一句是真的，一句都没有。我会说：“这根本不是真的引用。”它会回答：“是的，你对。那不是，这里有另一句。”我说：“这仍然不是真的引用。”所以，别向它要名言，或者如果你要了，一定要核实。它可能只是把某些文本总结得很有趣，看起来像引用，但并不是真的。

(01:15:06) Lenny Rachitsky

English:

That's a good reminder of just generally don't assume what it's telling is true. Generally, it's not giving you real things. It's making things up and usually they're right, but often they might not be right. That's a really good reminder. On that first tactic, is there a prompt that you find helpful for how to actually feed it in there? Is it just here's a line, help me come up with a better version of it?

中文翻译:

这是一个很好的提醒：总的来说，不要假设它说的是真的。通常它给你的不是真实的东西，它在编造。虽然通常是对的，但往往也可能不对。关于第一个战术，你有没有发现什么好用的提示词（prompt）？是直接说“这是一行字，帮我想个更好的版本”吗？

(01:15:26) Camille Fournier

English:

No, I have not figured out the prompt engineering thing at all. I tried to get it to summarize a paper for me the other day, and it literally gave me the summary of a different paper, and then I asked friends and they were like, "Well, here's a summary." I was like, "Actually, that seems like a good summary." And they're like, "Well, first I told the AI that it was an expert in the field and that it needed to read carefully and that it was really smart," and I'm just like, how do I have to manage the machine? I already have to manage all of these humans? Why are you making me manage machines now too, please?

中文翻译:

不，我完全没搞懂“提示词工程”（prompt engineering）。前几天我试着让它帮我总结一篇论文，它居然给了我另一篇论文的总结。然后我问朋友，他们给了我一个总结。我说：“实际上，这个总结看起来不错。”他们说：“哦，我先告诉 AI 它是一个该领域的专家，需要仔细阅读，而且它非常聪明。”我心想：我怎么还得管理机器？我已经要管理这么多人类了，为什么现在还要让我管理机器？求求了。

(01:16:00) Lenny Rachitsky

English:

I thought this was going to solve all of our problems. Yeah, that role. There's an acronym for how to approach engineering, and there's always like give it the role. Here's the role you're going to play for me. You are an amazing writer and here's what I want from you. Yeah. I'm also very bad at it. By the way, Claude I here is really good at writing. That's one of its superpowers, so if you want to try writing, that's worth trying. Claude by Anthropic. On the second point you made of fake quote. So, yeah, Francis Ford Coppola is coming out with this movie with ... As you mentioned, this trailer is just full of all these quotes that people supposedly said about all his other movies that Godfather and stuff. And they're all like, these are terrible ... They're all terribly mean quotes about his movies. It turns out, yeah, they're all not real quotes, and they actually took down the trailer, I just read because he's just making up quotes by famous critics.

中文翻译:

我本以为这能解决我们所有的问题。是的，设定角色。关于如何处理工程问题有一个缩写词，总是包含“给它一个角色”。“这是你要为我扮演的角色：你是一个了不起的作家，这是我想要的。”是的，我也不擅长这个。顺便说一句，我听说 Claude 的写作能力非常强，那是它的超能力之一。所以如果你想尝试写作，值得一试。Anthropic 出品的 Claude。关于你提到的虚假引用，是的，弗朗西斯·福特·科波拉的新片预告片里全是人们对他以前电影（如《教父》等）的评价。那些评价都很糟糕，对他电影的刻薄评价。结果证明，那些全不是真的引用。我刚读到他们撤下了预告片，因为他编造了著名影评人的话。

(01:16:49) Camille Fournier

English:

Well, so maybe-

中文翻译:

好吧，所以也许——

(01:16:50) Lenny Rachitsky

English:

Maybe it was ChatGPT.

中文翻译:

也许那是 ChatGPT 干的。

(01:16:54) Camille Fournier

English:

You can get fooled. I don't know if it was ChatGPT or a cheeky intern or something, but we can get-

中文翻译:

你可能会被骗。我不知道那是 ChatGPT 还是某个调皮的实习生干的，但我们可能会——

(01:16:58) Lenny Rachitsky

English:

Or very intentional marketing stint.

中文翻译:

或者是刻意的营销手段。

(01:17:04) Camille Fournier

English:

Or that.

中文翻译:

或者那样。

(01:17:04) Lenny Rachitsky

English:

Yeah. Anyway, with that Camille, we have reached our very exciting lightning round. Are you ready?

中文翻译:

是的。无论如何，Camille，我们已经进入了非常令人兴奋的闪电轮。准备好了吗?

(01:17:08) Camille Fournier

English:

Yes.

中文翻译:

准备好了。

(01:17:09) Lenny Rachitsky

English:

Amazing. Okay. First question, what are two or three books that you've recommended most to other people?

中文翻译:

太棒了。第一个问题：你向别人推荐最多的两三本书是什么？

(01:17:15) Camille Fournier

English:

So the first one is, What Got You Here Won't Get You There. I love it. I think anybody who is trying to challenge themselves and grow, should read it, it's fantastic. The second one is called When Things Fall Apart by Pema Chodron. That is when you are ... For me anyway, when I am struggling with whatever circumstances around me, I find it a very soothing or reminder that life is hard and the best way to ... you just sort of have to breathe with it and be with it and let it remind you of how life is hard for everyone and sort of make you a kinder person in the process.

中文翻译:

第一本是《魔劲》（What Got You Here Won't Get You There）。我非常喜欢它。我认为任何想要挑战自我并成长的人都应该读读，它太棒了。第二本是佩玛·丘卓（Pema Chodron）的《当生命陷落时》（When Things Fall Apart）。对我来说，当我为周围的环境感到挣扎时，我发现它非常抚慰人心，它提醒我生活是艰难的，而最好的方式就是……你只需要随之呼吸，与之共存，让它提醒你生活对每个人来说都不容易，并在这个过程中让你成为一个更善良的人。

(01:18:04) Lenny Rachitsky

English:

Is there a favorite recent movie or TV show you've really enjoyed?

中文翻译:

最近有没有什么你非常喜欢的电影或电视剧？

(01:18:08) Camille Fournier

English:

I loved Alien Romulus. It was great. If you like a scary alien movie, fantastic.

中文翻译:

我喜欢《异形：夺命舰》（Alien: Romulus）。非常棒。如果你喜欢恐怖的异形电影，那太棒了。

(01:18:15) Lenny Rachitsky

English:

I hate scary movies, so that's a new Alien movie. Okay.

中文翻译:

我讨厌恐怖片，所以那是部新的《异形》电影。好吧。

(01:18:18) Camille Fournier

English:

Yeah.

中文翻译:

是的。

(01:18:18) Lenny Rachitsky

English:

Anyone know there was a new Alien movie.

中文翻译:

有人知道出了一部新的《异形》电影。

(01:18:19) Camille Fournier

English:

I loved it.

中文翻译:

我很喜欢。

(01:18:21) Lenny Rachitsky

English:

Awesome. Is there a favorite product you've recently discovered that you really love, whether it's an app or even physical product?

中文翻译:

太棒了。最近有没有发现什么你非常喜欢的产品，无论是应用还是实物产品？

(01:18:27) Camille Fournier

English:

So I don't know about recent, I'm a big Whoop fan. I got it during the pandemic and it is one of my ... I'm just a weird fan girl for it. I don't know. I just really ... I enjoy it. I use it every day. Maybe it's totally inaccurate, but it seems accurate enough and I just find it a very interesting and cool product.

中文翻译:

我不确定算不算最近，我是 Whoop（健康追踪手环）的忠实粉丝。我在疫情期间买的，它是我的……我就是它的一个狂热小迷妹。我不知道，我真的很喜欢它，每天都用。也许它完全不准确，但看起来足够准确，我觉得它是一个非常有趣且酷的产品。

(01:18:48) Lenny Rachitsky

English:

I know Whoop product, people listen to this podcast, so I think they'll be happy to hear that. Two more questions. Do you have a favorite life motto that you often come back to, share with friends or family, find useful and work around life?

中文翻译:

我知道 Whoop 的产品人员也在听这个播客，我想他们听到这个会很高兴。还有两个问题。你有没有什么常用的人生格言，会分享给朋友或家人，或者觉得在工作和生活中很有用?

(01:18:59) Camille Fournier

English:

If you don't challenge yourself, if you don't take risks, you'll never grow. That's a big one. I think you've got to ... challenging yourself, taking risks is how you grow. I think the other one for me is stay curious. The more you can remember that there's more that you don't know than that you do know, and staying open-minded and being willing to be wrong, I think the happier you'll be and the better you'll be as a leader for sure.

中文翻译:

“如果你不挑战自己，不承担风险，你永远不会成长。”这是一个很重要的格言。我认为你必须……挑战自己、承担风险是成长的唯一途径。另一个格言是“保持好奇”。你越能记住你不知道的比你知道的多，保持开放的心态并愿意承认错误，你就会越快乐，也肯定会成为一个更好的领导者。

(01:19:29) Lenny Rachitsky

English:

Final question, on the topic of working out, you're blocking it with your head generally during the podcast, when we moved ... Behind you is a very significant looking barbell set or dumbbell set, I don't know which one is which. Talk about your workout regimen or anything along the lines of how you work out.

中文翻译:

最后一个问题，关于健身。录播客时你的头挡住了它，但当我们移动时……你身后有一套看起来很专业的杠铃或哑铃，我分不清是哪个。谈谈你的健身计划，或者你是如何健身的。

(01:19:48) Camille Fournier

English:

I have been lifting rates probably for longer than some of your podcast listeners have been alive, which says more about my age than anything else. So I used be ... I now have two kids, so I lift weights as sort of maintenance, but I used be a very ... I could deadlift than twice my body weight and I was just a very, very strong person. Now of course, everybody is into weightlifting, which makes the gym really annoying. So partly why I have a set in here, a small set with a baby sized bar is ... so when I can't make it to the gym, I can at least lift a little bit at home.

中文翻译:

我练举重的时间可能比你的一些听众出生的时间还要长，这更多说明了我的年龄。我以前……我现在有两个孩子，所以我举重是为了维持状态。但我以前非常……我能硬拉超过两倍体重的重量，我曾是一个非常强壮的人。现在当然每个人都迷上了举重，这让健身房变得很烦人。所以我在这里放了一套器材，一个带小杠铃杆的小套装，这样当我没法去健身房时，至少可以在家里练练。

(01:20:27) Camille Fournier (Continued)

English:

What's your favorite workout, lifting wise?

中文翻译:

在举重方面，你最喜欢的训练是什么？

(01:20:29) Camille Fournier

English:

I love really basic deadlift, squat, overhead press, that kind of very simple, the classic lifts.

中文翻译:

我喜欢非常基础的硬拉、深蹲、推举，就是那种非常简单、经典的动作。

(01:20:39) Lenny Rachitsky

English:

Amazing. Camille, this is awesome. As everything, I was hoping it'd be, we covered so much stuff. Two final questions. Where can folks find you online if they want to reach out, follow up on stuff and check out your books and how can listeners be useful to you?

中文翻译:

太棒了。Camille，这太精彩了。正如我所期待的，我们涵盖了这么多内容。最后两个问题：如果大家想联系你、跟进后续或查看你的书，可以在哪里找到你？听众可以为你做些什么？

(01:20:51) Camille Fournier

English:

Yes, so with the weird state of social media now, that's actually a harder question to answer than it normally is. I am on LinkedIn, so you can always follow me on LinkedIn and I'll probably ... I haven't traditionally put that much there, but I'm expecting I'll probably start putting more there in the coming months. I also still use Medium when I write. I actually like Medium, so medium.com, scamille is my username. Those are two good ways. I have a Twitter/X account that it is currently locked, and I may or may not unlock it at some point, but social media, as I said, has gotten weird. So I think probably LinkedIn is the easiest way for this kind of stuff.

中文翻译:

是的，鉴于现在社交媒体的古怪状态，这实际上比平时更难回答。我在 LinkedIn 上，所以你可以随时在 LinkedIn 上关注我。我以前在那儿发的东西不多，但我预计未来几个月会开始多发一些。我写文章时仍然使用 Medium，我挺喜欢 Medium 的，我的用户名是 scamille。这是两个好方法。我有一个 Twitter/X 账号，目前是锁定的，我可能会也可能不会在某个时候解锁。但正如我所说，社交媒体变得很奇怪。所以我觉得 LinkedIn 可能是最简单的方式。

(01:21:33) Lenny Rachitsky

English:

And I read somewhere that scamille is rooted in your love of ska music back when you were younger. Is that right?

中文翻译:

我在哪里读到过，scamille 这个名字源于你年轻时对 Ska 音乐的热爱。对吗?

(01:21:39) Camille Fournier

English:

It's true, yes. It was from high school. I was a big ska fan.

中文翻译:

是真的，是的。那是从高中开始的，我是 Ska 音乐的超级粉丝。

(01:21:43) Lenny Rachitsky

English:

It's funny how our usernames just stick from, we were really young and that's the way we're represented online now forever.

中文翻译:

很有趣，我们的用户名总是从很年轻的时候就固定下来，然后永远成为了我们在网上的代表。

(01:21:49) Camille Fournier

English:

Yeah.

中文翻译:

是的。

(01:21:50) Lenny Rachitsky

English:

That's absurd. And then, you didn't answer my final question, which is how can listeners be useful to you?

中文翻译:

这太荒谬了。然后，你还没回答我最后一个问题：听众可以为你做些什么？

(01:21:55) Camille Fournier

English:

I have a new book coming out. I have another book that I've already written. Obviously, I love it when people buy my books so I love when people share my books with other people. My first book is translated into a bunch of languages, which is super exciting, and I hope that if you read it and enjoy it and learn something from it, you will let me know. Tag me on some social media. I think that's just awesome and stay tuned. Look, if you are looking for people potentially to come talk to your company about platform engineering, I could be interested in that, but I always love to meet interesting people. If you're in New York, reach out, who knows. This has been an amazing, amazing time getting to chat with you, Lenny. Thank you so much for inviting me on the show.

中文翻译:

我有一本新书即将出版，还有一本已经写好的书。显然，我喜欢人们买我的书，也喜欢人们把我的书分享给别人。我的第一本书被翻译成了很多种语言，这非常令人兴奋。我希望如果你读了它、喜欢它并从中有所收获，请告诉我。在社交媒体上艾特我。我觉得那太棒了。另外，请保持关注。如果你正在寻找可以到你公司谈论平台工程的人，我可能会感兴趣。我总是喜欢结识有趣的人。如果你在纽约，联系我，谁知道呢。Lenny，和你聊天非常非常愉快。非常感谢你邀请我参加节目。

(01:22:42) Lenny Rachitsky

English:

I feel the same way. Thank you for agreeing to come on the show, Camille, you're awesome. Thank you so much for being here.

中文翻译:

我也有同感。感谢你同意参加节目，Camille，你太棒了。非常感谢你能来。

(01:22:48) Camille Fournier

English:

Take care.

中文翻译:

保重。

(01:22:49) Lenny Rachitsky

English:

Bye everyone. Thank you so much for listening. If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify or your favorite podcast app. Also, please consider giving us a rating or leaving a review as that really helps other listeners find the podcast. You can find all past episodes or learn more about the show at lennyspodcast.com. See you in the next episode.

中文翻译:

再见，各位。非常感谢收听。如果你觉得本期节目有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客应用中订阅。此外，请考虑给我们评分或留下评论，这能真正帮助其他听众发现本播客。你可以在 lennyspodcast.com 找到所有往期节目或了解更多信息。下期节目见。