Starting with what is your problem or what is your goal, I would say this is a bigger challenge than most people recognize or realize. 80% of the folks that I work with, this is their biggest problem, even at executive levels. Teams will have gone off for several months and they're tackling something and they'll come back with uncertainty and they'll say like,
will you told me to improve developer experience? I'm like, okay, what do you mean by this? Are you talking about inner and outer loop? Are you talking about friction? Are you talking about culture? But if you're talking about culture, this is totally different than if you're talking about friction in tool chains. If you're on different pages, you're heading in completely different directions. Welcome to Lenny's podcast where I interview world-class product leaders and growth experts to learn from their hard-won experiences building and growing today's most successful products. Today, my guest is Nicole Forsgren. This is actually my first recording back since going on Pat Lee for the past couple months and what an awesome episode to get back into the swing of things. Nicole is the developer productivity expert having written the award-winning book Accelerate and she's been the co-author of the State of DevOps report year after year. She's currently a partner at Microsoft Research leading developer productivity research and strategy and she's helped some of the biggest companies in the world move faster, improve product quality and transform their cultures. In her conversation, we get into the weeds of how to go about measuring and improving your engineering team's productivity and experience.
We talk about the Dora framework and the space framework and how to actually implement them to understand how your engineering team is doing. Nicole also shares benchmarks for what elite companies are at. We talk about why moving faster turns out to be one of the best ways to improve quality and stability plus pitfalls you want to avoid and also preview of a new book that she's working on and so much more. Enjoy this episode with Nicole Forsgren after a short word from our sponsors. Today's entire episode is brought to you by DX, a platform for measuring and improving developer productivity. DX is designed by the researchers behind frameworks such as Dora,
Space and DevX including Nicole Forsgren who is my guest for this very episode. If you've tried measuring developer productivity, you know that there are a lot of basic metrics out there and a lot of ways to do this wrong and getting that full view of productivity is still really hard.
DX tackles this problem by combining qualitative and quantitative insights based on the very research
Nicole and her team have done giving you full clarity into how your developers are doing.
DX is used by both startups and Fortune 500 companies including companies like Twilio, Amplitude, eBay, Brex, Toast, Pfizer and Procter & Gamble. To learn more about DX and get a demo of their product, visit their website at getdx.com slash Lenny. That's getdx.com slash Lenny.
Nicole, welcome to the podcast. Thank you so much. I'm excited to be here. I'm excited to have you here. I actually skip this question usually with guests, but I thought it'd be actually really valuable to spend a little time on your background. You have such a unique role and unique set of experiences. Could you just talk briefly about the things you've been up to in your career, where you've worked and then what you're up to now and what you focus on these days?
Sure. I appreciate the question because you're right. I sort of had this choose your own adventure background. I started as a software engineer at IBM. I was writing software for

large enterprise systems, which meant I ended up running them. I was also a sysadmin. I was rocking and stacking. I was running these really, really large labs. Then I stumbled into this seven-day march for several years and I was like, there has to be a better way. We're here in rumors of it, but management was not buying in. I decided to win this battle with data. I was like, I should go do a PhD. I ended up taking a slight pivot into PhD and management information systems, which some people are less familiar with, but it's basically a cross between tech and business. I ended up getting a fairly technical PhD. I went to a school. I went to University of Arizona, which has a very, very technical degree, but I liked that it crossed with business because then I had the ability to make these strong business case statements. How is or is the way that we develop and deliver software tied to outcomes at the individual level? Can I be more productive? Can I have better work-life balance? The team level is the team more productive, is the team more efficient, and the organizational level. This is what I was really interested in originally. Do I see better ROI? Do I see better efficiency? Because then I could sell it to people. That was really what I originally went into.
I was a professor for a handful of years because if you're doing research, traditionally, that's the job in academia. I also had a master's in accounting because that really helped me make that kind of like financial tie and understanding financial statements. Then after a handful of years, I walked away from tenure because academia was not convinced that DevOps was a thing. The DevOps wasn't real. Stay at a DevOps report, who I was doing with Dora, DevOps Research and Assessment in collaboration with Jez Humble and Gene Kim. We started that work with Puppet. Shout
out to Alana Brown for starting that and Nigel Kirsten and the team there. We kind of pivoted away and Chef, the little configuration management startup at the time, hired me and they're like, we'll give you half time to do research and half time to help our engineering practices improve. That's cool. Yeah. I mean, they were incredible because what startup is going to be like, yeah, do research. I was there for a year and a half and then left to do Dora full time. We actually had a SaaS offering. We continued this data DevOps report just under the door of the inner. We had a SaaS offering because so many large companies were like, I want my own customized measurement reading and report. Then the joke there, when we met with Gartner, they were like, your superpower here was that you tricked people into strategy, which was not only, how do I benchmark? That was kind of our top of the funnel because everyone wants to know how they compare. The important thing is what should you do next? What's the most important next step? How do I measure? What do I do next? That gave me this incredible view into advising large organizations into this transformation journey. Then we built out this amazing partner network because we weren't actually consulting. We just had the SaaS piece, but then how do you act on it? We were then acquired by Google. I was CEO and co-founder, so I led that acquisition and then integration and building up these teams in Google. After that point, I joined GitHub, which is the largest developer network. I had this amazing opportunity to do more grounded and applied research again. I was VP of Research and Strategy.
Then I went over to MSR, where I kind of wear a couple hats. Right now, I have a research lab there with an incredible team. It's the Developer Experience Lab, where we do a bunch of work across
productivity, community, and well-being. Then I also help with Microsoft's cross

company effort to improve their developer infrastructure. It's this round effort into how do I really remain engaged in measuring, applying, thinking about this work, both in very applied, concrete pieces, and incredibly forward-looking work with MSR.
Amazing. Just to clarify, MSR, is that Microsoft?
Yeah, thank you. MSR is Microsoft Research.
Okay, cool. You've shared a couple of these terms, DevOps, developer productivity. I'm curious what the term you like to use for this area you focus on, developer productivity, developer experience, DevOps, what's the best way to think about this?
I really love that you asked this question, because I think they're very related concepts that people sometimes conflate, but I see them as being different, so related but different. Productivity, I think, is basically how much we can get done and how much we can do over time. I think that's why it's so important to have this holistic measure, because we can't just brute force it. That's why when my team and I and a bunch of my peers study productivity, we include this community effect, because software is a team sport, we joke, and also why well-being is so important, because we see that when you do productivity the right way, we see sustainability, we see well-being, we see reductions in burnout.
Developer experience is very related and very tied to this, and it contributes to productivity, but developer experience is if you think about who your users are, developers really are your users in this software engineering, in the software development piece, and so it's developer experience is what is it like to write software? Is this a friction-free process? Is this a very predictable and certain experience? Can we reduce this uncertainty and increase the predictability here to contribute to productivity? Then how does DevOps fit into that just so that we have the mental model of these terms? People have co-opted the terms, and some people named their tools, DevOps. I'm maybe a little more old-school, so when I was doing a bunch of my DevOps research, it was the capabilities and tools and processes that we can use to improve our software development and delivery end-to-end, so that it's faster and it is more reliable. DevOps was this technical, architectural, cultural practices that enable us to do this work better, so that it is, yes, more productive. We have a better developer experience, so it was again this very holistic picture. What I love about this topic is that I've never met a founder or a leader who is not thinking, we need to move faster, we need our engineers to be more productive, we need to get things out, the door quicker. We want engineers to be happier. Nobody doesn't want that, and so that's why I'm excited to dig into a lot of these things. Is that roughly what you find as well, that nobody's ever like, we're good, we don't need any of this, we don't need to focus on this area? You know what? I'll say yes and, right? It kind of goes back to why I got into this, because on the one hand, you won't say anyone who's saying, we don't really need to go faster, everything's fine, but at the same time, very often, I will come into scenarios where I'll find myself in scenarios where people are like, I mean, it would be nice if we were going faster, but do we really need to? Show me the business case. What's the ROI? Or if we go too fast, we'll have an instability, right? What are our safety measures? Are we going to lose reliability? What is happening, right? And when I first started, like ITIL and ITSM, the old school kind of change management processes, the common knowledge was that you had to have at least a two-week wait for change approvals in order to get that stability. Turns out that's not, right? It was just kind of an old lifestyle, right? And so we kind of have this weird balance of,

I want to move faster, but is it worth the investment? What am I going to get for it? Are you sure this is the priority? Or I've been in meetings where it's like, oh, yes, absolutely, right? Like, this is a priority, but it's the lowest priority. And I'm like, right? So then what we want to do is we want to have these kind of pointed conversations or these kind of like socratic type questions and conversations where it's like, help me understand more what your concerns are. Are your concerns around reliability when you move faster? We're not just trying to like all the guardrails down and sprint for no purpose of sprinting. And this is where kind of the Dora and DevOps research program comes into play, where it's, we don't just want to move fast and take all guardrails down. We want to implement good technical practices, like automated testing, good architectural practices so that when you move fast, you are also more stable, right? We want to be thinking about improving the developer experience so that when we are faster, we are also saving time, right? And then we can highlight a handful of statistics like what is your typical time for feature delivery? What is your typical time to first PR? What is your typical time to steady state productivity? What is your typical time for code review and PR process? And if we are to do like back of the napkin math, what sorts of time are you spending here? And if we do a rough look at industry, what are your peers spending here? And are we losing time, right? And if we can turn this into a value calculation, what does that look like so that we can think about the priority and the strategy here? And I think that's where it becomes a more focused conversation. This is a great segue to something I was going to get to a little bit later, but let's just get into it, which is the Dora framework. And then there's also the space framework. Can you just talk about what these two are when you use one versus the other, and then how that essentially helps you measure and then improve productivity and developer experience? Sure, sure, absolutely. And I'm so glad you brought this up. So Dora is an entire research program. Now, many people, when they hear Dora now, they think of the four keys or the Dora four or the four metrics. And I think that's what the research program and the company ended up becoming most known for. And so that was the software delivery performance metrics. And those are, there's two speed and two stability metrics. So the speed metrics are lead time. So how long to take to get from code committed to code running and production, deployment frequency,
how often do you deploy code? And then the stability metrics are MTTR, meantime to restore. So if something happens, how long does it take you to come back and then change fail rate for every change that is pushed? What's the rough percentage of incidents that require human intervention? Now, the thing that was really interesting is when we started measuring these,
we found that they move in tandem now with very strong significance from a statistical standpoint. Now, what this means is now we say speed and stability move together. Most people only think about this from the speed standpoint, which means when you move faster, you are more stable, which means you're, you're pushing smaller changes more often, right? Because if you're pushing all the time, it's going to be very, very small changes, which means you have a smaller blast radius, which means when you push, you have an error in production. It's going to be easier to debug, right? It's going to be much easier to figure all of that out. Your meantime to restore and mitigate, it's going to be much faster. But that also means is the reverse, when you push changes less frequently, you will have more unstable systems because when you push less frequency, you will have very, very large batch changes, which means you will have a very high,

very large blast radius, which means when you do have a resulting bug error, you will have to disentangle this big ball of mud, right? And figure out which piece actually caused the error. Figure all of that out. That ended up being a big surprise, right? Because refer to my prior comment about, you know, ITIL and ITSM, if you're forcing a two week pause for change approvals, you're causing this batching up of changes. And sometimes people were waiting with two weeks is good, a month must be better, or three months must be better, or six months must be better. And I mean, just think about the merge conflicts you're causing, right? You're just causing so many challenges and figuring out how to push this code into production.
So many people think of those four metrics, one, because we found that speed and stability move together, and two, because we started publishing benchmarks on what this looks like for low, medium, high, and then elite performers for many times. This, I believe, may have been interesting. I'm not sure if it was useful or helpful, but I think it was interesting, because it gave people at least something to shoot for, something to aim for. I will definitely say, what's most important is knowing where you are and the progress that you're making, right? It doesn't matter if, frankly, you're a high performer or you're an elite performer, it matters that you know where you are and you're making progress, right? You know, can you push daily or on demand? Or is your only technical capability that you can push twice a year, right? Just know where you are, and is it a business decision or a technical capability? That's basically what it comes down to. I'm going to jump in real quick, just to highlight what you just said, which I think is extremely important and powerful, and people might kind of move on too quickly. I also want to ask you what the actual benchmarks are, if you can share those, whatever you want to share there, but before I ask that, essentially what you're sharing right now is just, I feel like the $64,000 question of this episode is just, how do I move faster as a team? And what I'm hearing is, essentially, it's ship smaller things is kind of at the core of it. And also, if quality is low, you're also saying the answer is ship more often, ship smaller things. Is that roughly the message? Yes, absolutely. It ends up being much, much safer. Amazing. So I think that's an extremely important takeaway that I think people would, I don't know, that's surprising to me to hear that it's quality comes from shipping faster, and then also to ship faster and move, help your team move faster, it's ship smaller things and just deploy more often. Yeah. Amazing. Okay, great. I know we'll talk more about this, but let me go back to the question I was going to ask is, are there benchmarks you can share just right now that you think would be useful to people? I know you said it was interesting and maybe not as useful to people as you mentioned. Yeah. So I will admit, I only have the 2019 benchmarks top of mind. The team of Google has continued that work since I left. It's been led by Dr. Dustin Smith. Nathan Harvey continues the work. So huge shout out to that team. Many others participate. You can go to dora.dev and find all of the continued reports. They've integrated all of this work, but I will say they've remained fairly consistent. So really quickly, I'll share the elite performance. So deployment frequency, you can deploy on demand, lead time for changes takes less than a day, time to restore is less than an hour, and your change failure rate is between zero and 15%. Amazing. Okay, I'm writing these down. These are extremely valuable.
And I will mention people will say, well, this is kind of like a chunk of time, right? It's not super precise. Precision isn't really super important here, right? Like I don't, it doesn't

really matter if you can, like if your lead time is, if it's less than a day, it's less than a day, right? Like that's fine from a business perspective. It doesn't matter if it's like four hours or like four hours and two minutes, right?
Right. General categories are fine. Now, I will say like the next category for lead time for changes by the day is if lead time is between a day and a week. And this is for, for good?
For high, yeah, between elite and high elite is less than a day and high is between a day and a week. And then it goes between a week and a month and between a month and six months, right? So so you can ask people and they can tell you, right? They can kind of hunch it.
And this is from committing code into the repo and it going out into production?
To it, to about like ring zero. So you don't like, don't worry if it's like, oh, well, now we need to think about like the global deploy and like, which is the final endpoint? It's like, how long does it take to get through your deployment pipeline? Because are you going to be surprised? Do you have fast feedback loops? How does your deployment pipeline work? Does your deployment pipeline work? Right? Or are you going to commit code? Are you going to wait for that final review for about three months? Is something going to happen or break? And when it comes back
to the developer, because something happened or broke, because that kind of happens, right? Are they going to have to insert themselves back in the code,
re-review all the things that happened three months ago? All so many other things happened. That's incredibly difficult, which to your prior question, this is how it relates to the developer experience. If something happened less than a day, and like it's a surprise and it's not great, but like whatever, right? Something happened downstream and I got to fix it. I'm still sitting in my code right in my head. I've got that mental model. I know what happened. Maybe it's not great,
but it's fine. If it happened three months ago and I get interrupted, first of all, interruptions like suck. That's not fun. Second of all, now I've got to like, re-remember, re-read all of this code, maybe reload an entire new workspace instead of libraries and everything. Because I, maybe it's a whole quarter ago and like, we thought we were done.
And I got to do the whole thing all over. If a listener is working at a startup,
I imagine they're hearing this and they're like, it takes a day to ship that, we ship it all day,
a thousand times a day. I imagine these benchmarks are more valuable for larger companies. Is there
a kind of buckets you think about for like, here's the size of company this is meant for,
and then do you think about anything differently for a startup, say, I don't know, 10 people?
If anyone is only listening to this, I just got the biggest smile because we saw no statistical significance between small companies and large companies. The only statistically significant difference was with retail. I'll come back to that. It's so funny because large companies would say, oh, but this isn't fair for us. We have more complex code bases. We have so many things to do. Small companies just don't have to deal with this. Small companies would come to me and they would
say, oh, but this isn't fair. Large companies have so much money. They have so many resources. They don't have to deal with all the things. This doesn't apply to me. So it's like, either way,
and on days when I was feeling real snarky, I'd be like, pick your excuse.

You've got your like, drop down. Now, when I say retail was a bit of an outlier, they had a statistically significant difference. Their difference was that they were actually better. Why? Now, I can't tell you why. I can, in a research paper, we'd have a discussion section, and this is where you get to guess. I would surmise, and we do this in the report, that it's probably because retail went through the retail apocalypse. If you didn't survive, like if you weren't just killing it, you did not survive. So many retail firms just did not make it through. You had to be at the top of your game. Black Friday, there's no such thing as not having systems that are performing incredibly well. There's no such thing as not being in the cloud, because if you cannot make it through bursting on demand, bursting like magic, sometimes I joke, right, you're not going to make it. And so I suspect, if I were to guess, if you're not already a high performer in the retail space, natural selection got rid of you.

That is really interesting. That makes a lot of sense. So I'm looking at these thresholds again, and I'm thinking from the perspective of a founder who's just like, I wish my engineering team moved faster. Essentially, you're saying if deploy times, if they deploy more than once a day, if their deploy frequency is on demand, or I think it was hourly, was kind of the other bucket, was that part of it? Yeah. And then their mean time, their fail rate is like less than 10%, and their mean time to recovery is less than an hour. Basically, you're doing great. That's kind of the message of this framework at least. And if you're not doing it through brute force and killing yourself, now, can I jump in here? Because then people are like, but how do I do this? So let's say that you're not in that category, and you're like, because this is the next, this is the piece of criticism I'll get about Dora. People are like, well, all you've done is make me feel bad. You gave me these metrics. You've judged me. Now I feel bad. And then I'm like, so there's Dora there. I wrote a book called Accelerate, which is like the first four years of the research compiled and put together and expanded in a few things. And I joke, there's a whole rest of the book. Dora is best known for the four metrics, but there's an entire research program supporting it. So it's not just these four metrics. What we find is that if you improve a set of capabilities, I loved your question around, what is DevOps? DevOps is not the tool chain you buy. Marketing teams label tool chains DevOps. They wanted your money. DevOps is a set of capabilities. They're technical capabilities. They're architectural capabilities. They're cultural capabilities. They are lean management practices that predict speed and stability. And then speed and stability gives you money, right? Because it's your ability to create these features that give you money. So when you work backwards, if you want money, you get the features fast. If you want the features fast and stable, you do the things. And the things are technical capabilities like automated testing, CICD. And CICD is continuous integration, continuous deployment, is that right? Yes. Trunk-based development, using a version control system. So do you have good technical practices? Do you have good architectural practices? Do you have a loosely coupled system? Are you using the cloud? Or if you're not in the cloud for whatever reason, are you using the underlying architectural pieces that enable good cloud to do the cloud right? Or if you're in the cloud and you're not realizing benefits because you're doing the cloud wrong, right? Do you have a good culture? So you don't just magically go fast and have stability, right? So working backwards, which pieces are you struggling at? Now,

you kind of noted down the benchmarks. If you go to dora.dev, the team at Google was lovely. We worked really closely with the team. They're keeping this updated. You can take a quick check. There's a button there that says quick check. You can plug in where you kind of think you are. Like I said, you can hunch it. And it'll tell you where you are in the benchmarks today. And what industry you're in. And then the cool part is it'll say, now, like you'll want to ask yourself, like, where am I struggling? But it'll say, for your performance profile, and for the industry that you're in, statistically over the last several years, these are probably your constraints. Okay. These are probably the things that you're struggling in right now, right? Like, for people in finance who are high performers, they tend to struggle with these four things, right? Whether it's like culture or continuous integration or whatever. I love that you're getting tactical with how to actually improve these already, which is the bread and butter of this podcast. And so we'll link to this quick check. It's just dora.dev slash quick check. And by the way, they do not collect your name. They do not collect your info. There is no, there's no lead, any lead gen anything. Everything's just there. And then there's deep dives into every single one of the capabilities.
Amazing. And also your book talks about all these things. So people should go check out the book, obviously. It's on Amazon, Search Accelerate. Is that right? Yeah. Okay. So we were talking about dora. This may be a good time to talk about space, which I think is a different framework you recommend. What is that all about? Okay. So space is a way to measure, we say productivity, developer productivity, but it's a little bit more than that. Space is a good way to measure any type of complex creative work. Now, how do they relate? Let's say you go through the quick check. It points out like four things and you decide you want to improve. Continues integration and culture, right? Well, now you're like, cool, but how am I going to actually measure them? This is where space comes in. Because space helps you figure out, space gives you a framework to pick the right metrics. Now, some people are like, well, space, you didn't give me the exact metrics. People love dora because it's like, here's the exact four you need. Well, space is like when you want to measure something that's complex, creative work, maybe like developer
productivity, there's also an example at the bottom for incident management. When you have something you want to measure, it says within your context, within the metrics you have available to you, here's how to pick. That's what space is good for. Now, we called it space because it stands for the five dimensions that you want to measure. So S is satisfaction and well-being. So satisfaction well-being is kind of self-explanatory. Now, some people might jump in here and say, oh, well, you're just, you know, touchy feeling. This actually matters because we find that satisfaction well-being ends up being incredibly highly correlated with all of the other dimensions of productivity and doing things well. And as soon as satisfaction and well-being, things like sustainability, if you're satisfied, as soon as that starts falling off, other things start to break. So this can be an incredibly strong and important signal. P is performance. This is going to be the outcome of a process. So reliability within dora, the MTTM or change fail rate, right? Those are both performance metrics. And so you pick one to kind of measure as performance. Yep. A is activity. Anytime you have a count or a number of something, these we see all the time because they're super easy to instrument and automate, right? Number of pull requests, number of check-ins, number of something, that's A. C is communication

collaboration.
This can be how people work and talk together. It can be meetings. It can be collaboration.
It can also be how our systems communicate together. It can be the searchability of a
code base. And then E is efficiency and flow. So this is going to be the flow through the system.
It can be the time through the system. If we think about SRE or incident management,
it can be the number of hops a ticket takes until it reaches the right person.
Now, to use space correctly, we want to use at least three dimensions at a time
because that helps us balance. Turns out Dora is actually an implementation of space.
So Dora would be space for mostly that outer loop. So again, once you found something that you want
to improve, find the metrics that make sense to you. Try to have them be in balance or intention
so you don't throw something at a whack. But pick three. So when you say Dora is an
implementation of space, one has five buckets, one has four. How do you actually think about that?
So space is there to help you think about how you want to pick metrics. So a lot of time I see
people, so we have to step back. I used to advise people on how to pick metrics. For years,
people would pull me in to advise on Dora or accelerate. They would ask me questions,
but it ended up being metrics questions a lot. How do I pick the right metrics
to improve what I'm doing? Like I said, they had the Dora numbers.
They would pick their constraints and they wanted to improve. But how do I improve?
How do I measure this? How do I show improvement? And so we would start thinking really critically
about which metrics were the right metrics to pick. And I would always say, make sure you pick
balanced metrics. Make sure you pick metrics that are intentioned.
And I could say it, but people have a hard time wrapping that around their heads because they
kept picking things like, number of lines of code, never picked number of lines of code,
number of, still every month I get an email like this, number of pull requests, number of commits.
And I was like, these are all activity metrics. And so finally I pulled a few of my friends
together and I was like, let's come up with a framework to help people think about it.
And so there are five broad categories. Pick three, because that will help force you through
the mental exercise of what could I possibly pick? You don't need all five, right? This isn't,
we're not playing bingo. We're not playing blackout bingo. You don't need all of them.
But try to have at least three across different dimensions. Now, one example here,
I was working with a group that wanted to improve their pull requests very generally.
They just said improve pull requests. So they were thinking about pinging someone every 15 minutes.
And I was like, oh, this is going to be bad. Because we know from other literature and research,
like nursing, you'll get alert fatigue where people will just start tuning out alerts.
Either they'll turn them off or they will just stop hearing them. So like number of alerts,
they're like, let's just think about number of alerts. And I said, well, but if we think about
efficiency and flow, how much time do you have to work on your coding? So those two are balanced.
So we need to protect time to work, as well as code review time, pull request time.
And so sometimes we can think about those. And then I think we added a satisfaction metric.
Are you satisfied with the pull request process and the selection of the reviewer?
How do you go about actually capturing and measuring the say satisfaction?

So for satisfaction, I would generally ask. Go ahead and ask. Now, the ones that you instrument, you can instrument and pull out of systems all the time. Go ahead and grab that string.
For a satisfaction metric, I would only pull that periodically once every few months.
Like a survey to your engineering team.
Like a survey. Yep, absolutely. Awesome.
Don't discount what people say. Sometimes I hear, actually not sometimes,
a lot of times I hear people say, oh, but people lie. First of all, what is their incentive to lie?
Why would they lie about having a bad system? Because it's bad and they want it fixed.
Right? If it's absolutely a hostile work environment,
they might lie and then tell you it's good, then you have bigger problems.
Right? Also, do we ever see bad data from our systems or incomplete data from our systems?
That's a lie. But we find ways to deal with it. We see it. We acknowledge it.
We look for holes in our system data. We try to deal with it. Right? That's also a lie.
So I think there are better ways to think about and deal with that
and then try to work with it. Right? Because then I wrote a paper with Mick Kirsten on this several years ago on how data from people and data from systems are really important compliments because we can get certain insights from people that we'll never get from systems. Right? Like let's look at lead time from changes, for example. Right? From commit to deploy.
The speed might be fine, but people might tell you it's taking absolute heroics.
Right? It's some ridiculous Rube Goldberg machine. The system will never tell you that.
Or you could get data on your version control system. I worked with a company several years ago and we found out that there was a significant portion of code that was just
not going into any version control system. You're never going to find that out from your systems because it's not in the systems and it was mission critical.
I can see why people come to you asking for advice on metrics because you have this framework of here's the type of metrics you want and then I think and especially from an engineering team there's going to be this like how do I optimize and make sure I'm doing the right thing and measuring the right things. For someone that wants to do this and an hour-long podcast isn't going to give them all the answers, would you recommend they go read or go do or look at to help them figure that out? One, I hate to be this person, but I'll point to a few of my papers because I will say I write things down because I get asked them so often and I want to make sure it is broadly applicable or broadly available I guess. This space paper for sure. It's an ACM and I think the year we published it it was like the most read paper at ACMQ. We tried to make it as readable as possible so the space paper is nice because it outlines this framework and it gives examples of metrics in every single category and so hopefully people can look at this and they can say okay here's an example to use here. Here are some of the things that I could possibly use and we're seeing that space is being used lots and lots of different places. Another good one could be the paper that I mentioned with Mick Kirsten and it was about we talked about using data from people and data from systems. We wrote it up in the DevOps context because I want to say this is written in like 2016 or 2017 or something, but it helps you think through what types of data are good in which situations, right? Because you will never find yourself in a situation when you don't want both types of data. Even teams that I've worked with that are the most advanced, they have absolute instrumentation in every possible scenario.

In the most detailed way, they will still survey their developers at least once a year because you can get new insights, right? One book that I love, it's a little dense but it's really interesting that I love is How to Measure Anything and it's by Hubbard and there are parts of it that are like real stats heavy, but he has this portion in the front that's like covering intangibles and so it's like what happens when you don't have data? You have no data, you're starting from nothing, what are good ways to hunch data? I really love that because he covers some really good ground there. Today's entire episode is brought to you by DX, a platform for measuring and improving developer productivity. DX is designed by the researchers behind frameworks
such as Dora, Space and DevX, including Nicole Forsgren, who is my guest for this very episode. If you've tried measuring developer productivity, you know that there are a lot of basic metrics out there and a lot of ways to do this wrong and getting that full view of productivity is still really hard. DX tackles this problem by combining qualitative and quantitative insights based on the very research Nicole and her team have done, giving you full clarity into how your developers are doing. DX is used by both startups and Fortune 500 companies, including companies like Twilio, Amplitude, eBay, Brex, Toast, Pfizer and Procter & Gamble. To learn more about DX and get a demo of their product, visit their website at getdx.com. That's getdx.com. You also mentioned offline that you might be working on a book that will answer a lot of these questions. Is that something you're up for chatting about? Yeah, absolutely. So as I mentioned, I tend to write things down when I get asked questions on it a lot. And so this is one in particular. So we'll be covering, you know, I'm starting to go through and I'm covering some of these. And I think some of the important topics in particular are, you know, starting with what is your problem or what is your goal and being super, super crisp on it, right? Like, what is it that we're trying to answer? And I would say this is a bigger challenge than most people recognize or realize. Like, I'm making this setup, right? Like 80% of the folks that I work with, this, this is their biggest problem. Even at like executive levels, they'll, they'll ask their team or teams will come back with uncertainty and they'll say like, will you told me to improve developer experience? And like, okay, great. What do you mean by that? And then teams will have gone off for several months
and they're tackling something and they'll come back and they'll be like, oh, that wasn't what I meant. And I'm like, okay, what do you mean by this? Are you talking about inner and outer loop? Are you talking about friction? Are you talking about culture? Because sometimes they're talking about culture. And if you're talking about culture, this is an incredibly valid answer. But if you're talking about culture, this is totally different than if you're talking about friction in tool chains, right? And if you're on different pages, you're heading in completely different directions. So like that, that's one thing we cover, which seems obvious, but trust me, it is not. And then even like, how do you, we're going to do kind of a rough version of like, how do you start measuring from nothing? And also the measurement journey, right? Like, how do you think about the trade-offs between and the proportion of measurement between subjective data, right? Data from people. So you interviews and you have surveys and objective data, stuff you get from systems. Because when you first start off, you'll be relying much more on data from people because you can get it relatively quickly. But as you kind of transition through this measurement journey, you'll get more and

more data from your systems because it's scalable, it can be engineered, you can be doing, you know,
much more with it. And also you should be thinking about, you know, don't let the perfect be the enemy, the good, right? So like, how do we think about this very, very strategically? How do we transition through this? How do we think about what each piece of data is for? And also like, lots and lots of examples, right? So I have included like, example interview scripts. How do you select people? How do you screen people? Example survey scripts? What are some of the analyses we should do? And trying to make this incredibly accessible. So like, basically anyone can do this. So you do not need to be a data scientist. But if you have one on staff,
like you can hand them some of this and just like, let them run.
I think this book is going to do extremely well. I definitely come back on when it is out. I think you said maybe year-ish kind of timeframe? Yeah, probably about a year by the time we get all the way through. If people want to be notified when it's out, can they sign up on your site for a newsletter or anything like that? Or is there any way to kind of be in the loop as it approaches? Oh, yeah, absolutely. Yeah, I'll add a link for that. Also, if anyone is doing some of this work now, if they have major questions that they would love to have me to answer, if they have success stories, if they have case studies, if they have anything that they would love to be included,
I remember when I wrote Accelerate Before, there were a couple of folks that reached out after and they would like, oh, I wanted to have something included. Now, today I've learned, right? If there's
anything that folks would love to be in discussion with me about, I'm always eager to chat and nerd out about DevX and especially measurement journeys. Awesome. I usually ask this at the end and I have
more questions, but while we're here, how would people reach out to you? What's the best way to contact me? On my website, I've got info.nicolefi at Gmail. Awesome. Okay, a few more questions. Awesome. Thanks. What are the most common pitfalls that companies run into when they're trying to
roll out any sort of developer experience, developer productivity, system measurements, improvements? I think one, I just mentioned, right? Like not being clear or not understanding what it is that they're looking for, because then you can have a thousand flowers bloom and everyone's kind of running in a different direction. I think another one is not pursuing
this in both a top down and a bottom up structure. And I think that can really help
drive success and having good communication throughout is super, super important, right? So getting your ICs bought in and helping them understand that this is for them,
we want to understand what they're doing, knowing what vocabulary they use, what terminology they
use is super important, and then chatting with leaders, right? And understanding what their motivations are or helping them understand what the motivations could be. This kind of harkens back to one of our earliest chats on why I even got into this and how I see two different sides to the conversation on like, why is DevOps even a thing? Why should we even ship faster? There are so many people that I talk to that are super passionate about DevX right now,
and they're like, how can I convince my executive team this is important because
their developers are just completely burning out or they use computers in anger every day.

And so it's like, how can we have the right tools to socialize this to our leaders as well, right? Because this should be a priority. This needs to be a strategic piece. And how can we help pull together the right value points to communicate this and to understand what their priorities are so that we can see how this fits in, right? You've been working in this space for a long time, probably longer than anyone that has ever worked on this area of developer experience productivity. What have you seen change most from the time you started working in the space to today? What kind of progress has been made? We have these increasingly large complex systems, right? So like 10 or 15 years ago, like the internet was around, but like, things were really different. Now, almost every company has a really large complex system, right? We also have a shortage of developers, or at least a reported perceived shortage of developers. More companies are technology driven, or at least they understand they're technology driven. It's like, I remember a handful of years ago when I met with a financial institution whose CTO insisted to me that he was not a tech company. That's not real anymore. That doesn't happen anymore, at least like very, very rarely. So all of these things come together. And suddenly, many more companies are like, we have to be better at this. And that was not always the case, like five to 10 years ago. I used to have to really explain why this was oppressing concern and why it would continue to be a pressing concern. And now, in the last six to nine to 12 months, we have this AI moment happening. And it just poured gas on top of everything. Because now what's important, like we've always said that like ideas aren't important, execution is important. But now, this is absolutely true. Because it's not just about what it is that you build. It's about creating absolutely novel, incredibly new experiences, and doing them at a speed that no one has seen before. And the only way to do this is to have this software pipeline that is fast, and is safe, and is stable, and is reliable. And that's where we're seeing this really interesting convergence and pressure isn't quite the right word, but it's really forcing the discussion, and strategy and prioritization, right? I'm glad you touched on AI. That was actually exactly where I was going to go next. Perfect. Yeah, obviously, productivity, AI, engineering, something that's top of mind for a lot of people. There's a lot of layoffs that have been happening. There's a lot of talk of, we don't need as many engineers. I actually had dinner not too long ago with a few, I'd say, 10x engineers. And those are folks that people sometimes say like, they don't need Copilot, they're not going to use any of these tools, they're already amazing. And they were the opposite, they're like, this is making me 100% more effective and efficient than I love it. So clearly, good things are happening there. I don't know what the question is specifically, but I guess, have you seen the impact of AI on engineering productivity? And has that shifted how you think about developer experience and productivity beyond what you already just shared? Absolutely. Yes, and, right, I think this is a super interesting open question. So can I answer it just with a whole bunch of questions? Absolutely. We're absolutely seeing an impact, and we continue to explore this. So I have an interesting question to see how it'll change the space framework. What's open here? I think a few things will remain. Satisfaction is still going to be there, performance is still going to be there, activity is still going to be there, how you communicate with people and with the tool, efficiency and flow is still going to be there. I believe it will change an added dimension like trust or reliability. Can I rely on it? Well, I have an over-reliance on it, and what we're seeing is that, probably unsurprisingly,

people really fundamentally shift the way they
work when they work with an AI-enabled tool, like GitHub Copilot or Tab 9 or others,
because now instead of just writing code or having a short autocomplete,
you spend more time reviewing code than writing code. There's this wonderful paper
out that uses the CUPS model. I'll share it with you. A team at MSR did it. It finds that
about 50% of your time now is spent reviewing versus writing.
But it'll be interesting to see how that changes things longitudinally,
because some of my colleagues also did a paper that showed that
you can do certain tasks like build an HTTP server 50% faster. But I don't think that's what
productivity is about when you're using an AI tool, frankly. Anyone who's looking at that,
and dear CEOs or whoever who are like, now I can lay off at my workforce,
that's not what this is about. It's not about taking a task and cutting your time in half,
because now what we've enabled is your ability to do certain things faster,
and then free up some of your cognitive space so that you can do harder things with this new
co-pilot sidecar or something. But also, because now you're accepting text and then reviewing it,
we've changed what your mental model is. So we've changed the friction model that you expect,
we've changed the cognitive load of what you expect. We're changing reliance on code. So what
does this mean for reliance or over reliance? What does this mean for learning? What does this
mean for novices versus experts? How do we measure productivity? There are a handful of us
that are having these discussions on what does this mean, and how do we communicate it
thoughtfully?
Again, we really need to have these holistic balanced metrics, because if it's an oversimplification,
we really risk losing the forest for the trees. But it's also super interesting and super compelling,
how can we think about learning or onboarding to new code bases or new languages for folks who
already know computational learning? I think it's also very different for folks who are just learning
programming languages and don't already know things like computational thinking.
If someone was excited to go down this road of we're going to focus on developer experience,
we're going to focus on helping your engineers be more productive, what are the next
step or two that they should take in your opinion just broadly knowing that you don't know any
specifics about the company that's thinking about this right now? I think if you're walking away
from this podcast and you're like, I'm already working on this or I think this is a thing that's
happening, I would say just go check your work basically. Has this been written down? Is there
a clearly defined challenge, problem, something? Start there. Absolutely. Because that is going
to be the thing that reduces confusion, the best. Absolutely. And then see if there's any data.
And data can be very loosely defined. Is there any signal that is related to the problem?
Like I'd start there. And you can do that. You can do that a week. You can hunt something down.
Sounds like something you could do in a day. Yeah. Well, depending on how scattered things are.
Are there any companies that you look at as good models of they do this really well?
I think Google does this incredibly well. And sometimes I hesitate to mention Google because
they're like, you know, some people are like, well, we can't be Google and we aren't really
advanced. But the thing I love about Google's approach is that they've really taken kind of
this measurement phase approach to things, right, even when they roll it out in new places.
They're very systematic in how they measure things. They have incredible telemetry and

tooling and instrumentation. And they continue to invest time in developer experience surveys, and they triangulate them. And one thing that I also love being able to point out here is if there is ever a disagreement between the surveys and the instrumentation, which is incredibly advanced almost every time, every time that I've ever heard of the surveys are correct and not the instrumentation. Amazing. I have just a couple more questions unrelated to this topic. Is there anything else that you thought you think would be useful to share or leave people with around this general space? I would say that like thinking about what it is you want to do is always important, right? Like getting crisp, the ability to communicate clearly is always one of the best things. One of, I think, one of my superpowers and one of the things that I've been working with my teams on doing and kind of teaching them is, and one of the things that's really leveled up our work in general is making your work incredibly accessible. And accessible, not necessarily like the accessibility definition of the word, but making it very easy to understand what you're doing for your key audiences. And so thinking about doing that for anything that, you know, anyone who's listening for all of your work is super important, right? So who is it that your audience is? What's their role? What words resonate with them? And then always being able to translate your work into a few sentences or a paragraph or less. I love it. A lot of the listeners of this podcast are product managers. And this is so core to the work of a PM. So I think this is perfect. Speaking really directly to a lot of the listeners. Okay, so just a couple more questions. Before this podcast, I asked you a few questions, including just like, what are people asking you for advice often around? And are there any other frameworks that you find really useful? And so there's a couple things I just wanted to touch on, see if there's something interesting there.
The first is you have this framework that you call the four box framework. I'm curious what that is and what it's all about. Yes, I love this four box framework. I've used it for years. I actually pulled it out first when I was a professor. And I still, to this day, get LinkedIn messages from my students saying that it's like the most useful thing they've ever used. So here's what it is.
I literally pulled this out on napkins at bars at conferences to this day. So here we go. Draw four boxes on a piece of paper, two on the top, two on the bottom. So they'll be kind of aligned.
The first two, to the left of them, write the word words.
And below them, write the word data. And then between the two on the top, draw an arrow between them. So it'll say words, box, arrow, box, right? Does that make any sense? Then on the bottom, it'll say data, box, arrow, box. Okay. So on the top half, this is where if you want to think about measuring something or testing something, you have to start with words. So as an example, let's just say, I think that customer satisfaction gets us more money.
Or customer satisfaction gets us return customers. Let's do customer satisfaction.
So the first box, you'll put customer satisfaction inside the box, and you'll put return customers in the second box. Now, always start with words. Do not start with data. You always start with words.
And then you'll go around to a couple people, stakeholders, managers, others, and you'll say, do you agree with this? Is this actually what we're doing? It can turn into a sentence.
And then in the boxes below it, this is your data. How are we going to measure customer satisfaction?
It could be a survey. And so like this is where you'll go and you'll say, like, what data points do we have that could proxy for what could be our data points for customer satisfaction? And this is where it gets tricky because you could say, well, customer satisfaction could be return customers,

but we think it leads to return customers. So we can't use that here. But return customers or could be so like, that's where you kind of like roll this out. So how else would we measure customer satisfaction? I made this hard on myself. Like a CSAT score or CSAT?
Yeah, CSAT, NPS, we could say the amount of money that they spent. It's a stretch.
Okay, now return customers. Let's go to the next box. How are we going to measure return customers?
Depending on our context, let's say that this is an online business, we could say that it's return customers as measured through the website. We could say that it's return customers, we could just ask them, right? Maybe we have a follow-up survey, return customers, maybe we're going to do a stretch here, maybe we say it's a referral link. This helps us get super clear on what it is we're going to measure. Now, the reason I like this is because if some of our data, now this data analysis, we'll just do correlations here, right? If we have longitudinal over time, that's fine.
You can hand this to like a data scientist. You can hand this to someone and you can say, what data do we have? Let's go run this. If something here falls apart, now you can point to the data boxes and we can get mad about the things in the data boxes and we can say, what's wrong? Is the data poor quality? Are we missing data? Was this a bad proxy, right? Proxy stands for something else. Was this ridiculous, right? One of the things I made up, right? It was just a bad idea. Instead of getting mad at Lenny for his really stupid idea or getting mad at Nicole because this was a really bad idea, we can say, this was problematic. What's wrong here? Or we can go back up to the words at the top and we can say, this is not actually something that is probably going to hold or this is not something we want to test right now or this is something. It makes things incredibly clear. It helps you communicate what it is you want to do fairly quickly.
I love it. Here's mine. Check it out. It's ugly. I'll zoom in right now. I will say advanced mode. You can start with the same four box framework and you can say, what data do we have available? What do we think the relationships are? But then you have to go back up to words and then say, for these data points and we think that they represent something and we think this is the relationship between them, what do they represent? If I turn this into a sentence, what do they represent? Then you want to double check because Spurrius Correlation is one of my favorite websites
instead of charts. You'll want to go chat with someone, interview, make sure things are actually right. But the challenge is I will see people run every correlation they could think of, but they haven't turned it into a word or a sentence that you can communicate to someone else. They don't do the check and they don't do that before, one, before running the correlations. Two, if it's there, all of our data is so interrelated that we quite often will find Spurrius Correlations. But it can be really helpful just to have that laid out, even if it's just not a post it, to say, what are the things I expect to see? What is this actually testing? What relationship do I suspect is there? Amazing. I have a newsletter guest post on how to do a correlation analysis and regression analysis so folks can read that. Awesome. Oh, that's so great. Plug and play, all kinds of it makes it easy for you. So what I'm going to take you away from this is this is an awesome framework, especially for thinking about a hypothesis you may have. In this case, it's like customer satisfaction is going to lead to more return customers. Here's how we're going to measure it. And then you basically run the desk and see if it's true. And if it's not, maybe you need to pick different metrics, maybe you need to pick a different

conclusion. And within the Dora framework, we would say if we want to improve our speed and stability, we think improving build time would help. And then how would I measure build time, right? These are the data points that I have available to us. Yep, to circle back.
I love it. It's all connected. Okay. And then last question. I asked you what advice people often ask you for. And you said that it's around making decisions. And I'm curious, what advice do you give people about making decisions? Yes. So this one comes up in business, but also comes up personally. And among my mentees. So many times it starts with, you know, being very crisp about your objectives and definitions. But then it comes down to, you know, really clearly defining what your criteria is, what's important. And then among that criteria, what's most important. Some of my friends know I have a decision-making spreadsheet that I have shared out with a handful of friends on, you know, should you take a job? Where should you move? What are the different things really useful? It is, it's well, it's funny though, because what's interesting is many times I will like, I'll share it with someone and I've got a couple that are just funny, right?
But walking through the spreadsheet is often all you need to do in order to know what the decision is. And by that, I mean, so we walked through the decision, I had one where I was like, where should I move next? Or like, what job should I take? Right? So when I started Dora, I did this. Starting Dora, I thought was my lowest. Once I walked through this spreadsheet, it became my high. So what you do is you outline, like, of all of your options, what do you want to do? And then you say, what are the criteria that are important to me? So if it's for a job, is it something like total comp, cash money, prestige, team, job predictability, work-life balance, identify the criteria that are most important to you. Now, it's really interesting because sometimes I will only get that far when I'm working with someone I'm mentoring or coaching, and they will say, I know what my answer is. We don't even get to the next step. But just identifying the criteria that are important, is it? Now, when I was thinking about where I wanted to move next, it was proximity to an airport, the relative tech scene, the food scene, that was real high for me, a handful of things, that was important. Now, the next thing I do is for each criteria, what's their relative weight? What's their importance? And I make it add up 200%. And then I, like, this is the easy part, right? Like, you just put it in a little spreadsheet and I then I give everything to score and I just multiply it out. Now, this is where I'm data informed and I'm not data driven. There have been times I make a decision where, you know,
the whole, like, flip a coin and, like, whatever it's, wherever it lands on, what your reaction is tells you what it should actually be. There have been times where, like, I multiply it out and then I'll actually, like, fudge the numbers to get what I want, but it's still slightly off. That's where your data informed. Same thing in business. There are many times where you actually run the numbers and it'll give you a class or a category of things and then you choose. Now, this is where, you know, one of my favorite quotes I heard somewhere about strategy comes into play, right? And that's that the key to having a good strategy is knowing what not to do and the key to executing a good strategy is actually not doing it.
So you can have many options, right? As a leader and as an executive, we have many options and we only fund some of them. If you fund everything, things are going to fail.
So being able to think through and identify what your criteria are, identifying that criteria, what's your selection criteria, what's your evaluative criteria, ranking them and then

deciding what the cutoff is, is important. You can't fund everything. You don't get to pick everything. Amazing. I love the spreadsheet idea. I've made versions of it, but it's always, I think like you said a lot of times, the exercise is just tell you what you already think and just gives you like, yeah, all right, you're right. You probably should just do that thing you already thought you should do. Yep. Have you thought about making a public template of this spreadsheet even though it is simple? I bet it would be really helpful to people.
I have and this actually might be a good forcing function, maybe.
Okay. Awesome. So if you do it, I'll put in the show notes, it'll probably be near the bottom at the end of the episode, but that'd be awesome. Perfect. Is there anything else that you want to share before we get to our very exciting lightning round? No, I think that's it. Well, welcome to our very exciting lightning round. I've got six questions for you. Are you ready?
Absolutely. All right. First question, what are two or three books that you've recommended most to other people? We actually had the perfect segue because the book I've recommended absolutely the most is called Good Strategy, Bad Strategy by Richard Rumelt. Another one is Designing Your Life
by Bill Burnett and Dave Evans. And the last one is probably Ender's Game. Orson Scott Card.
No comment right now on some of his political commentary, but I used to have extra copies in my office when I was a professor and I would just hand it out to my students.
It's a fun, like, just easy nonsense read, but... I absolutely love it. Such a good pick.
Haven't read in a long time. And are they making a show of that at all? That'd be something.
They made a movie and I was afraid I wasn't going to like it, so I just didn't read it because I didn't want it to ruin the book. But at least Harrison Ford was in it. Okay, I'm not going to check it out. They're making a movie of three body problem. I don't know if you've read that, but I'm really excited for that. It's on my list. Oh, man. Best sci-fi ever.
Next question, actually. Very correlated. What is a favorite recent movie or TV show?
I think going through some real, just easy fun watches lately. I'm re-watching Suits again, but Ted Lasso is a favorite and I just tore through Never Have I Ever, which is fun because John McEnroe narrates it, which is hilarious. John McEnroe, the tennis player?
Yeah, it's a riot. Yeah, it's so funny. I love it. Next question. What's
a favorite interview question that you like to ask people when you're interviewing them?
I love questions that I can kind of spin around hard decisions that people have had to make and how they made them. I love hearing their thought process.
And I get a little nervous when people just like yolo and shoot from the hip constantly.
So what is it you look for there that gives you a sense that they're
someone you may want to hire or work with? I just like hearing if they have some sort of process. If they have some kind of decision-making process, if they have criteria, if they have how do they do evaluation. What is a favorite product that you've recently discovered that you love? I have a big one, a little one. My big one is probably Sleep 8. So I live in Arizona.
It gets hot here sometimes. Oh, 8 Sleep. Yeah, 8 Sleep. Yeah, the other way around.
Yeah, so that one's fun because it makes the bed cold and also gives me some data, which is probably like a little bit off, but in the approximation it's fun.
And then Korean face masks, they're just fun. Yeah, you can get some pretty good ones for like just a couple dollars and that's always fun. Self-care. First mention of that one, Korean face

masks. Right. Listen, everyone get on board. I just did the TikTok. There's a filter now where you could see how you look when you age and I'm not happy with how it turned out. And so I might look into this. I had some basal cell cancer on my forehead a few years ago. And so I am much more careful with my skin. And you can get like one of my favorites is Cosrx. You can get 10 for like $15. So it's fun to just like chill at the end of the day with a good face mask.
I was going to ask you for a specific pick and so we got one. Yep. Amazing. This next question, I ask everyone and it's especially appropriate to you, but I don't know if you'll have an answer. What's something relatively minor you've changed in your product development process that has had a big impact on your team's ability to execute? And I feel like you have a big perspective on this. So I'm curious what you have as an answer.
I think I alluded to this earlier. I would say that it's, you know, helping everyone. So I've done this before, but I think it's helping everyone to ask who's our audience and how will we share this now? And it's sort of interesting because right now I'm wearing two hats. One is it MSR or Microsoft Research. We need very ambitious research, right? So like H2H3.
What is H3? The third cap? Oh, Horizon 3. And so it's supposed to be like 5 to 10 years out, which right now is like who even knows, right? We're going to be in computers.
Yeah. AI has like completely upended how we kind of think of horizons. And so when we're thinking really ambitiously and very, very, very forward looking, what's our check in? How do we evaluate this? And then how can we easily communicate it to our core audience? And so here, who's our audience and how do we bring the far near?
And then for the other hat I'm wearing, I'm working with Okto kind of across all of Microsoft to take a data-informed approach to really improve and up-level our central developer infrastructure. And so as we're thinking very, very tactically, what is our long-term vision and how do we align with several of our broad stakeholders? And so there it's who's our audience and how do we bring the near far? I love that. Final question. What is one tactical piece of advice that listeners can do this week to help improve their developer productivity or developer experience and move it in the right direction? You know, if you walk away from this podcast right now, you could take a look at what's happening in your org today. Is it written down? Is it clear? Do you have any existing data and efforts? And if not, go find a handful of developers and ask them how they feel about their work tools and their work process and what the biggest barriers
to their productivity are. Also pick up a copy of Accelerate on Amazon or your local retail establishment. Nicole, this was amazing. I think we're going to help a lot of companies move faster, have better and happier engineers, which is going to create infinite value in the world.
Thank you so much for being here. Two final questions. Where can folks find you online if they want to reach out? And how can listeners be useful to you? I'm on Twitter and Blue Sky, NicoleFV. And my website is NicoleFV.com. And all my contact information is there.
And as we mentioned previously, I'm working on a new project and a new book digging into exactly these ideas, right? How can we measure better? How can we improve? And what does that measurement
process look like? Both for one-time, really quick, unofficial measurement pieces and if we want to do very formal, longer-term measurement pieces. So if anyone is interested in that or has any success stories they'd love to share, I would love to hear more about it. So please

reach out and share. I'd love to hear more. Awesome. Nicole, thank you again for being here. Thank you, Lenny. 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 Lenny'sPodcast.com. See you in the next episode.