EPISODE 964

[INTRODUCTION]

[00:00:00] JM: Kubernetes has created a widespread system for deploying and managing infrastructure. As Kubernetes has been increasingly adapted, companies are thinking about how to leverage that common layer of infrastructure. With the common infrastructure abstraction of Kubernetes, it becomes easier to adopt other abstractions that are uniform across the entire company, and this has created a market opportunity for products such as a service mesh.

A service mesh consists of sidecar containers that get deployed alongside services in a distributed system. These sidecar containers often get deployed in the same pod as the other Kubernetes containers. A pod is something that contains multiple containers or just one container. Each sidecar container is used as a proxy for all the communications that go through the service that is deployed with. This consistent proxying layer provides each service with benefits such as security, and routing, and telemetry, and policy management, and we've done many previous shows about service mesh.

Istio is a service mesh that was created and open sourced by Google. Istio is built around the Envoy service proxy sidecar and a control plane that manages the Envoy sidecars. Since the launch of Istio, some of the Google employees who were working on Istio have started Tetrate, a company with the goal of commercializing Istio into a product that enterprises will pay for.

The market demand for service mesh has been proven, but there are many competitors to Tetrate. Istio is open source and can be commercialized by other companies as well as cloud providers such as Google and AWS. Linkerd is a service mesh built by the company Buoyant, which was the first company to focus exclusively on this space. There are other companies that are expanding existing products into becoming a service mesh. These are companies like Kong, and NGINX, and HashiCorp.

Zach Butcher is a founding engineer with Tetrate and he joins the show to discuss the market for service mesh and the plan for Tetrate to build a business around Istio.

Quick announcement; we are hiring for two roles, a content writer and an operations lead. If you like to write about software engineering and you have some familiarity with software engineering, maybe you're a computer science student, maybe you're an experienced engineer, send me an email, jeff@softwareengineeringdaily.com, and also the operations lead role is for somebody who's interested in learning more about how to run a business, how to run a podcast and who wants to help us improve our operations. You can also send me an email if you're just in that, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:02:54] JM: I love software architecture. Software architecture is the high-level perspective of how to build software systems. Much of Software Engineering Daily is about software architecture, and if you're interested in software architecture, there's no better place to go to discuss and learn about software architecture than the O'Reilly Software Architecture Conference, which is coming to New York February 23rd through 26th of 2020.

If you are interested in software architecture, you can go to oreillysacon.com/sedaily. That link is in the show notes, and you can get 20% off your ticket to the software architecture conference. The O'Reilly Software Architecture Conference is a great place to learn about the high-level perspectives and the implementation details of microservices, cloud computing, serverless and also systems like machine learning and analytics.

If you've been listening to Software Engineering Daily for a while, you know that these systems are hard to build and they take engineering details at both the high-level and at the low-level. Whether you're a seasoned architect or an engineer that is just curious about software architecture and maybe you want to become a software architect, you can check out the O'Reilly Software Architecture Conference at oreillysacon.com/sedaily. Use the discount code SE20 and get 20% off your ticket.

There are lots of reasons to go to the software architecture conference. There's networking opportunities. There are plenty of talks and training opportunities, and you can get 20% off by going to oreillysacon.com/sedaily and entering discount code SE20. I've been going to O'Reilly conferences for years and I don't see myself stopping anytime soon, because they're just a

great way to learn and meet people. So check it out, and thanks to O'Reilly for being a longtime sponsor of Software Engineering Daily.

[INTERVIEW]

[00:05:12] JM: Zach Butcher, welcome to Software Engineering Daily.

[00:05:14] ZB: Thank you. Thanks for having me. I'm excited to be here.

[00:05:16] JM: I want to start with the subject of microservices, because we're at KubeCon, and I know that most people here believe that microservices is something that everyone should work towards having. The idea of putting a lot of effort into building a microservices architecture, this has a cost. So if you put a bunch of effort into re-platforming your entire architecture into Kubernetes, in a CI/CD workflow that's kind of complicated and you adapt a service smash. If you want to do all that –

[00:05:52] ZB: It's a lot of complexity. Yeah.

[00:05:54] JM: Not only is lot of complexity. It's an opportunity cost, because you're giving up building business logic, and customers may or may not want a microservices platform. They probably don't even care. All they want is more functionality, and re-platforming doesn't really help with that. Why should we spend so much time or who should be spending time building our a microservices architecture?

[00:06:19] ZB: Yeah. So just like any technology, you don't want to go – In our industry, as software engineers, we really like to grab the new shiny toys, because they're new and shiny and exciting, right? It doesn't really work for a business, right?

So as we look at what actually matters with respect to this decision, am I going to take my monolithic or whatever my current architecture is that I've been running that I hopefully am pretty comfortable running and fundamentally change it and add in probably a lot of complexity. It's a really tough thing to weigh.

By and large, the overriding factor that we've seen from our customers and just talking across the industry is this kind of idea of development agility, right? We want to go faster is kind of the calling card, and the idea is this; my customers want features. They don't care how I'm running my infrastructure. They don't care what I – But they want to do more things in my software today. So I try and get my developers to do more things in the software today, but they wind up because of the way that we've decided to deploy our software is like one bundled unit together and maybe we haven't invested in things like high-level traffic control. Maybe we're only doing – We don't have fine-grained control be able to do things like canaries or gradually deploy new traffic.

So changes are kind of risky, because this big blob that 10 or 20 different teams have contributed to, this big monolithic thing, has to go out. I know I made some changes from my part of it. I'm sure some of the other teams made changes to their parts, but hopefully it all plays well. So this becomes a very risky thing, right? Really, the result is some ossification, right? I can't – Because it's risky to change and its – My customers want new features, but they want to be able to use the product first.

So if I'm having outages, if I'm not able to serve my products because I am updating, and updates are risky and they go bad, then I'm not in a good state. So I have to be able to de-risk change. I need to be able to decouple my teams from each other so that they can move and operate independently, right? Not everybody needs to change at the same rate. But some pieces of the product probably need to undergo a very rapid rate of change, and you don't want them to be gated on everybody else, and you don't want to force everybody else to go up-tempo because of the one team.

So this is really where the idea of splitting these components apart at a deployment level, literally taking the code and splitting it into separate pieces that run separately and that communicate with each other, and because you're splitting them apart now – So that is the fundamental challenge, right? We need to split them apart so that teams can go faster, and that introduces effectively all of the complexity that KubeCon here we're talking about Istio, the project we'll talk about in a minute that I work on that helps to address the fact that suddenly now because I've decoupled my programs to decouple my teams, the network is a fundamental part of my application now.

[00:09:22] JM: So this is a nice story. I've definitely heard it before, and I know it has been applied successfully at Google, at Netflix. Maybe at Uber, although I think Uber has – I spoke to an engineer there who is a little bit remorseful about the microservices decision. I mean, I honestly don't know really what the alternative is to microservices architecture. I mean, it's maybe like being comfortable with having five or six giant monoliths and then other smaller services.

[00:09:53] ZB: Yeah, the reality is I think it's some mix. So Larry Peterson is the guy that is the CTO of the Open Networking Foundation, and they're responsible for a bunch of telecom networking standards, but they also played a really big part, for example, in the development of software defined networking, and that is one of the key enabling technologies of cloud, really.

He, I think, actually gets a kind of a little bit of what you're talking about in a very succinct way, which is he talks about we disaggregate to innovate, but we have to re-aggregate to operationalize. This starts to get some of this kind of core idea of why do we have to do this process. We see this repeated across the industry.

As software engineers, we complain about the cyclic nature of our industry, right? We complain that Kafka looks a lot like service buses and we learn that enterprise service buses are bad. So why are we back into this architecture again? The answer is that we're not really back into the same architecture. We've learn and made changes along the way. We go through this cycle where we have a system. We really only know how to operate holistic systems. That's why monoliths are so nice, because they're one piece. So they're easier to understand and operate.

When I disaggregate it, when I break it apart, I can change it more rapidly, but it's so hard to operate. We're really, really bad. When I say we, I mean like across the board, not just talking about microservices, but in general in software engineering and, we're not as good at dealing with disaggregated pieces than we are with holes, right? We're used to lumping things together so that we can mentally treat them in one way.

So this is kind of some of that fundamental tension, I think, that we see in our software development is this cycle of, "Well, we want to microservices and we had to do that, because

we had to innovate, because we had to disaggregate so that individual pieces could move faster," but now it hurts. So you talk to a software engineer who goes, "I don't like the service mesh. I don't like this micro service idea, because the operational tools aren't there."

We can barely debug concurrent programs on one computer, and now you're saying we're going to split it apart in different computers and they're going to communicate over the network to do this stuff. It's a real tooling problem in part, and some of that acute pain at least I think that we hear about in – That's that knee-jerk reaction to the architecture, right?

[00:12:09] JM: What I'm trying to understand is how successful has this microservices mass migration been? Because I feel like there's some survivorship bias. We come to KubeCon and we see five or six talks about Lyft's journey to microservices, or ZenDesk, breaking up our monolith into the microservices till we have 99% of our services each occupying 1% of the overall infrastructure and we've got it evenly distributed. Perfectly load balanced. You never hear the stories of we spent 2-1/2 years trying to make this thing work and ended up going back to COBOL.

[00:12:56] ZB: Yeah, totally. I can say from like the perspective of the products that Tetrate is building and shipping, I'm building a monolithic binary right now, because it's easier to operate, right? Again, the operational side of it is easier. I haven't hit the pain points that would necessitate needing to split up into many different binaries yet, right? My teams are still able to ship features independently of each other at a rate that is fast enough that we feel comfortable. Things like that. But from an operational perspective, it's just easier.

I think that that's the case for a lot of people, right? I really do – The by and large use case for microservices I think really is a small number of companies, a small – It's those ones that are larger. Again, it's ones that are large enough to have the organizational pain of how do we get these teams to operate together, right? It's really not a technical problem. Microservices do not solve any – They do not really solve any technical problem that you have, and instead they introduce a whole lot of them, but they solve organizational problems that you have. That's why people are moving to them. The organizational side of it, right?

The technical side is the cost that we have to overcome to enable the organization, right? There're a lot of ideas around – You led off with what are some different architectures, for example, that people are looking at, right? This is where there is a lot of work in this space.

So we were talking before we started recording, we mentioned just a little bit about Knative and like open PaaS, for example. So the whole serverless paradigm is one reaction to this pain, right? It goes maybe even more extreme, and we're going to take our units of code that we cut up and we're going to cut them up even smaller. That's not necessarily required for a serverless. So that's one reaction that people are having.

Another reaction is the return of the monolith, right? Again, I view this as part of that cycle of disaggregation and re-aggregation, right? It's not necessarily a bad thing that we go back to a monolithic deployment if we can maintain the advantage of having decoupled development teams. It turns out that there's, for example, sets of techniques that we can use to do this, right? So I can have everybody develop independent services, but maybe stitch them together into a one binary in the main and make it communicate locally rather than over a network, right?

Now I have some of the operational benefits of having a monolithic deployment, but I can maybe get – Also win some of the benefits of having my teams decoupled from each other if we can work out that sticky mess around how we do deployments and how we keep the rate of change fast enough for the entire organization, right?

[00:15:29] JM: You're coming at this from a pretty interesting perspective, because you worked at Google. You were working on Istio in the early days. You worked with Varun, who I've interviewed on the show a couple of times, and eventually he left and you left. You guys were both part of the founding of Tetrate, which works on service mesh and related technologies.

The advantage of being on your own in a startup is you are forced to go and talk to customers and really extract the truth, because it's existential. It's existential for you as a startup. Whereas if you're at a big company, like I'm sure you guys were trying hard, but you don't necessarily need to get the straight answers because your salary is going to be there whether or not.

[00:16:18] ZB: Yeah, that's one of my – I mean, this is one of the fundamental takeaways about Tetrate, the company, coming from Google, because both Varun and I came from Google, and he's been there for a decade, right? I'd only been there for three and half years, but I've been there for a while. But you go talk to an Amazon engineer and almost every single sentence has the word customer in it that they say, right? If you go talk to a Google engineer, you never hear the word customer ever, and this was one of my –

[00:16:44] JM: Until you have the ad side of the house.

[00:16:47] ZB: Well, yes. And the one side of the house that is customer-oriented is ads, and they're good at it, and the rest of the organization is not, because that's fundamentally not what they've been building –

[00:16:57] JM: Their customer is the engineer.

[00:16:58] ZB: Exactly, right? So the AI –

[00:17:01] JM: And selling directly to engineers that live inside of your organization who are not paying you money is very different than selling to enterprises.

[00:17:08] ZB: Precisely. So this was one of my single biggest segues personally leaving Google. Google is a magnificent place to go learn a whole bunch of different things. But for me personally this was the most important thing that I took away, was that Tetrate had to be a customer-focused company. It is an existential problem exactly like you say, right?

[00:17:28] JM: So when you go out and have those conversations, when you talk to these enterprises and you say, "Look, I know there's a lot of technological change going on right now in your infrastructure. You're looking at Kubernetes. You're looking at three gigantic cloud providers and a bunch of adjunct cloud providers. You're looking at a bajillion little vendors that are selling you monitoring software and logging software and this software and that software, and then you into security and you got another 500 grand that you're going to need spend.

[00:17:57] ZB: That's some pretty cheap security you're getting.

[00:17:58] JM: Okay. Sure. Yeah. 500 grand, that's not very realistic. But your specific category is kind of new, the idea of service mesh. When you talk to them and you say like, "What do you actually want? We are a "service mesh" company, but maybe we can do other things. What do you want from us?" What do they say? By the way, what kinds of enterprises are we talking about? Are we talking about banks? Are we talking about insurance companies? What are we talking about?

[00:18:24] ZB: Yeah. So our customer base is predominantly financial tech companies, in particular, payment institutions.

[00:18:30] JM: Gigantic, old companies. Lots of money to spend. Good reason to spend it.

[00:18:34] ZB: Yup, exactly. So those are the groups that we talk with, and by and large when we go talk to them, they need a couple of different things. Traditionally, the team that we interface with is actually a new one. So exactly to your point. So, historically, these three verticals, the security, networking and observability have been independent and they've been handled independently across the organization, right? Actually, to the pain of the organization most the time, right? How much do you go talk to people in companies with more [inaudible 00:19:04] workflows and you don't have to get around the security team to get approvals and they're a big roadblock and it's like, "Oh! I can't tell you how many times I've talked with customers, and they talk about the security team with dread, because they're the people that say no."

To go get a change in the network – So if I'm a service owner and I'm just trying to get my feature to my user, I have to go get the security team to approve my new thing. I had to go get the networking team to go make my changes. I have to go talk with the observability team to set up all these stuff, right? So these historically have been siloed, and arguably to the detriment of the business.

So one of the key changes that we're seeing with a lot of the people that we're interacting with is that a new team is starting to be created in these organizations, which is the platform team, if you will. So now the group has been chartered in a lot of these large organizations whose

purview is go make these things coherent." They'll figure out, "Hey, that cloud thing is happening." "Hey, this Kubernetes thing is happening." "Hey, we have these data centers now that are in VM's. We realized that we need to modernize. We realized that our competitors are investing in modernization. They're building out. They're becoming technology companies in some sense, in payments, right?"

Right now in the financial space across the board, across most of the verticals in finance, it's a race to become a technology company in some sense, right? So they're realizing we need more agility, we need to be able to go faster. We need to be able to do these things. So they're starting to build this platform team, and this tends to be a new team that is relatively recently chartered, and they're given this purview to interact across these pillars to start to try and figure out how we make a coherent platform for our developers, right?

So when I go in and talk with them, typically – Obviously, because of the market, the industry, this queue is towards security. In particular, one of the really interesting things is the idea of application level identity, right? So today when we talk about security postures and things like that, we tend to talk about network-based security, right? This application sits in this subnet and we've allocated them this /whatever – They get these eight IP addresses. Those are those nodes. We're going to open up the physical firewall to talk to its database, which I know is dedicated to these four IP addresses.

Maybe it's a little bit more coarse-grained than that, but by and large it is this – We have physical firewalls connecting subnets together, and this is not amenable with a cloud world. This is not amenable with Kubernetes, where your network – Where your identifier, your network address, can change.

One of the key features that's really interesting is we start try and bridge these heterogeneous environments as we're trying to figure out how do I run workloads in cloud and on-prem together and make them talk. Application level identity, having your job present a token that you can authenticate and that you can authorize against that you trust the application what it is and you don't have to trust the network is critical, is a key feature for them. That's one of the knock-on benefit of that if you're going to assign identity, the way that Istio does it in particular implies that

you can do encryption in transit as well. So those identities that we give are in the form of certificates.

So the another benefit is for a variety of regulatory requirements. For regulatory reasons, you need encryption in transit, right? So those two things, “Hey, I can get identity. That gives me policy that I can write that starts to become dependent of the network,” and this is very new for security teams. So part of this is having the security teams start to realize that they can start a phrase policy in better ways or in ways that are more expressive.

[00:22:41] JM: So let me see if I understand you correctly. So the security product that people are asking for is a way of assigning identity to applications, first of all, and I think that's encompassed in the spiffy inspire project also

[00:23:00] ZB: Totally. It is too. Yeah, exactly. Istio used spiffy, for example, the spec. Spiffy is the specifications, fire the implementation. Istio also implements spiffy to do identities, for example.

[00:23:10] JM: Right. Okay. Great. It's basically a spec for here's how you assign an identifier to an application that can be used throughout your infrastructure for various things, such as security policy. You need to assign certificates to those applications so that they can do TLS handshakes, right?

[00:23:31] ZB: Yeah, exactly. So this is where like – So, Spire, what Spire does and what the certificate rotation side of the Istio control plane does is take care of giving you those certificates that have your identities inside.

[00:23:43] JM: So when I talk to the console service mash people, basically like Console, they took a key value system and kind of rebranded it as a service mesh. But basically they said that the reason that they did that was because when they talk to people, like HashiCorp, they’re super smart. They're figuring out what do people actually want from a service mesh. Service mesh sounds like something we should do. We’re like the kind of nebulous, but unicorn –

[00:24:11] ZB: They're able to do it, right? They have the technical chops to be able to sell it legitimately. They're one of those few companies that they can come out with one and people would believe it.

[00:24:19] JM: Exactly. Exactly. They have technical jobs. People respect them. Yeah, it is perfect for them. But when they looked at like, "Okay. There's all these things that these service meshes are doing." When you go and talk to the Istio people or you go and talk to the Linkerd people. There's this laundry list of things they do, the load balancing, A-B testing, green-blue deployments, red-white deployments, black-orange deployments. This and that and sliced bread. Ultimately, what people want is security. What does security boiled down to? Policy management and application identity.

[00:24:56] ZB: So maybe one of the dirty secrets of networking is that networking has always been a security sell, right? I mean, VMware arguably is – The big thing that they did was make software defined networking a mainstay for people, right? How did they do it? Micro-segmentation security.

Networking has always been a security sell to some degree. Now I do want to say though, you asked what's the driving use case for my set of customers, right? So what we hear from them is I want all of them, but is what hurts most now.

[00:25:23] JM: This is at the top. Right. Okay.

[00:25:24] ZB: Right. This is actually I think a pretty key point too, because adaption is really hard. One of the keys for adaption is pick one, exactly one pain point, right? Because basically the delta, the increase in complexity to add a feature, not that big. But the increase in complexity to start using it for the first time, big. So you need something that's sufficiently painful to overcome that initial adaption pain. Once you've done that initial adaption, the incremental addition of features is pretty easy, because the delta to learn is pretty small.

[00:25:57] JM: So that platform team. Let's say the platform team at an insurance company or a bank that's developing. Their mission is to figure out infrastructure that fits across the entire

organization that they can sort of slot into uniformly across the organization?

[00:26:16] ZB: To some degree. Maybe the charter is not so much uniformity as figuring out how we're going to develop all new software and then figure out how we're going to take what is legacy and bring it into the new world.

[00:26:28] JM: Okay. So they're saying for all greenfield applications, we want to have some standards. So the greenfield applications don't have the problems of the legacy stuff.

[00:26:35] ZB: Yeah. So like, "Hey! You're not going to deploy your Greenfield into a VM. We're on to Kubernetes now." So new applications go there, right? For example.

[00:26:43] JM: And then as they prove that out, maybe they can apply it to older applications too.

[00:26:46] ZB: Exactly, because there's a strong desire. Right. It's not like just because the teams run on legacy land that they're not totally happy there, right? They would like to be able to go faster. They would like some of the features of the service mesh, for example.

It's really funny to me, a lot of times we'll go in and talk with customers and one of the things we'll show is like some of the observably side and they go, "No! We have observability. We have some sort of vendor name."

[00:27:09] JM: [inaudible 00:27:09].

[00:27:10] ZB: No, but like insert a telemetry vendor, right? SignalFX, whoever. We're not really interested in that. Then you saw, "Oh! But we can actually have high-level metrics." They go, "Whoa! I didn't know that that's even a thing that we can have, and now I can give it to my default." So it's a really exciting – It's an exciting thing.

[00:27:27] JM: Right. Okay. So they have this platform team, and the platform team can win over the security team and have standards going forward, and the security stuff that you give them, how hard is it to create a service mesh or a platform system for deploying stuff that has

those security properties, the policy management properties that they want. Is this a reality today or is this something you're working towards?

[00:28:02] ZB: So it's a reality in some environments, and we're working towards another. Part of this is just the security teams themselves need to convince themselves a new model, right? This application- based identity is fundamentally a different model for implementing policy and security, right? There's a whole lot of complexity in there, right? So as soon as you start issuing these identities, you run into the problems of how do I authenticate the workload? How do I know that I'm issuing the right identity to the right thing? There're all these knock on problems, right? That's why Sitel exists. That's why Spire is a product that – And all of that. It's a really challenging problem. So it's in a state of flux, right?

So security teams would be negligent if they dropped their existing security policy and went whole hog into this new thing. What regulator? That would be crazy to do from like a regulatory standpoint, for example. Because this is a new uncharted world where – But an auditor knows the controls for traditional network security.

So the reality is that today it's starts to be a mix, right? A lot of the things that we wind up discussing with a lot of these is how do we do things like in the new world? We use application identity from Spiffy in some form with this [inaudible 00:29:13] Spire or something like that. But when I go talk back to legacy land, we need to do a swap of identity so that I can integrate in with the legacy view of the security model, right? So that's one of the transitory states that people –

[00:29:24] JM: What is that? Is that like a translation layer or something additional?

[00:29:28] ZB: That's one of those things that winds up being pretty like organization specific, because your network is one of those things that solve – It's always a special snowflake, right? I've seen a bunch of different things from different users that take – Some of them, it's simple things like VPNs or NATing and stuff like that. Some of them, it's more sophisticated. We're going to go in through ingresses and we're going to do "real", but we're going to do as if it's an end-user. We're going to treat them as end-user, clients calling in, and the whole spectrum between, right?

I guess your original question though was how real is it today?

[00:30:01] JM: Yeah.

[00:30:01] ZB: So these policies exist today, you can offer them. It's fine. We enforce them at runtime. That's great, and really the hurdle becomes getting the security team to buy-in to an updated model.

[SPONSOR MESSAGE]

[00:30:19] JM: Cox Automotive is the technology company behind Kelly Blue Book, autotrader.com and many other car sales and information platforms. Cox Automotive transforms the way that the world buys, sells and owns cars. They have the data and the user base to understand the future of car purchasing and ownership.

Cox automotive is looking for software engineers, data engineers, Scrum masters and a variety of other positions to help push the technology forward. If you want to innovate in the world of car buying, selling and ownership, check out cox autotech.com. That's C-O-X-A-U-T-O-T-E-C-H.com to find out more about career opportunities and what it's like working at Cox Automotive.

Cox Automotive isn't a car company. They're a technology company that's transforming the automotive industry.

Thanks to Cox Automotive, and if you want to support the show and check out the job opportunities at Cox Automotive, go to coxautotech.com.

[INTERVIEW CONTINUED]

[00:31:36] JM: The platform solution that you're selling these people, or advising them on, or consulting with them on, are they using Kubernetes and Istio and like sidecar containers?

[00:31:51] ZB: Yeah. Yeah, typically. So most of these platforms teams that we talked to want to use Kubernetes for the new platform, right?

[00:31:59] JM: The Greenfield application.

[00:31:59] ZB: Yeah, the greenfield, and all the ones that we have already have substantial workloads in Kubernetes because it was greenfield two or three years ago, right? So they already have a pretty big split footprint between those two. I'm sorry. What was the first part of your question?

[00:32:14] JM: Well, no. That answered it. I mean, I was basically kind of hinting back at the beginning of the conversation like have these old legacy enterprises that have tons and tons of infrastructure and tons and tons of greenfield mileage ahead of them, are they going into "microservices"? It sounds like they are, and it's a reality.

[00:32:32] ZB: Yeah, for sure. So those companies are exactly the ones that need the organizational –So those companies have historically been the ones that are pretty ossified, that are hard to change, right? So again they see that it's an existential problem for them. They have to become nimbler. They have to become technology companies in some sense, right? Companies like Square and Stripe are phenomenal examples that put the fear and they look at – So payment processing companies on the East Coast look over at those and they go, "I need to – That's what I'm playing against."

So they look at it from a holistic perspective, "I need to skill up my developers." Part of it is just even hiring side of it, right? To some degree, if I really want some of the best people, I need to be using some of the cool technology, because the best people want to do that, right? It's a hiring tool. It's a combination, right? It's this holistic, "From the perspective of my company, I need to make this transition, and there's many different pieces that belong to that, right?" This actually addresses – This move to this adaption of cloud native technology, let's move to Kubernetes. These things enable that transition, that strategic goal, across a bunch of different dimensions; hiring, culture.

[00:33:46] JM: Absolutely. I think again at this point. I mean, you're making a very strong case for the idea that microservices is something that you go for because of organizational reasons, and then it creates technical difficulties, but the technical difficulties are worth overcoming, because you will get a stronger organization out of it and ultimately better technology out of it, because of the better organization.

Very specific question, let's say a company adapted Kubernetes three years ago. The footprint has been growing since then. Some enterprising set of engineers got Kubernetes of the ground. They started saying like, “Look, here's our platform plan. We got this platform plan for Kubernetes,” and everybody throughout the organization going forward has implemented new applications using Kubernetes, using containers.

Let's say three years into this, this company starts to say, “We want to start having a sidecar proxy. We want to start having Envoy sidecar proxy, and then eventually we want to have Istio. So we want have Istio in addition to the Envoy sidecars that are going to be proxying all the traffic between each other.” How hard is it to deploy sidecar containers throughout that kind of organization?

[00:34:58] ZB: Yeah. So this is actually one kind of the key pain points of Istio, right? It's one of these things that we still have not really addressed fully, which is the day zero is still kind of tough.

[00:35:10] JM: Day zero of Envoy.

[00:35:11] ZB: Of Istio in particular. Envoy too. Just be clear for listeners. So Envoy is a component in Istio. Envoy itself open source project very successful is used by a bunch of different service mesh implementations, is used by a bunch of people building their own bespoke “mesh”, because the degree to which it’s a mesh or not depends on their organization. But a lot of people are using this is a proxy independently.

Istio basically provides the batteries to a program on voice, right? So your original question was how hard is it?

[00:35:42] JM: Was how hard is it? Specifically, thinking about the Envoy deployment process. But I guess most of the people who are deploying Envoy, it's in service of Istio.

[00:35:50] ZB: Yeah. Well, and a lot of them are deploying it themselves. Depending on your – If you're in Kubernetes, like you said, for example, that's pretty easy, right? Because the primitives are already in place to be able to do in a platform, right?

[00:36:01] JM: So just throw them in Istio.

[00:36:03] ZB: Right. I put the container in my pod spec. Hey, it turns out the pod spec can have many containers. You just put a second one, right? That's the mechanics. The challenge becomes is the semantics right? Have I programed Istio to be able to work with my service correctly and not break it? That's the pain point today with Istio.

Maybe the most classic historic example has been port naming. I want my – I would like to be able to proxy HTTP traffic in my application. It turns out that if you want to do that in Istio, historically, we have always required that you explicitly name your Kubernetes port HTTP–something. Don't care what it is. But we use that as a signal.

So one of the classic pain points was, "Hey, I wanted to try Istio. I went ahead and just deployed the sidecar on all my services and none of the traffic works." We go, "Oh! Did you label all the ports?" "No. I didn't know I need to."

So there's these tripping hazards, right? So one of the most acute pain points for the project for the past two years or so has been these day zero, day one pain points around how do we actually enable adaption, right? Unfortunately, it's taking quite a lot of time to get there, but especially from 1.4, which just came out last week, Istio 1.4 and then some of the – And 1.5 should have even more changes, that start to alleviate a lot of these introductory problems, a lot of these stumbling blocks that you would typically hit on these initial adaption journeys, right?

But it turns out, it is quite a bit of work to adapt this, right? Again, I think I said earlier, the most successful adaptors I see pick exactly one pain point. The most successful adaptors I see additionally start gradually, right? You don't just turn it on everywhere. You pick your victim or

your volunteer carefully first, right? And you do this gradually. This is how – But this is really how any large change in any organization happens, right? So it's not really – That's not a novel, right?

[00:38:08] JM: The example I always remember is the classic Netflix migration to microservices, which began with the jobs board, the Netflix jobs board. Netflix had a monolithic architecture, and the first thing that they – Oh! No, I'm sorry. This is Netflix moving to the cloud. But –

[00:38:25] ZB: Yeah. It's the same idea. Yeah.

[00:38:27] JM: Same idea, this because the same idea is any big technological shift, you start with something that has low service area and low risk to the organization. If your jobs board goes down, it doesn't matter. If you want to deploy a service mesh, you probably start with the jobs board also. Who cares? You get the jobs board up and running. You get the – Whatever. Test the number of jobs board instances you can set up. Are they all observable through Istio? Can you change their security policy? Oops! We made a mistake. The jobs board is offline for five minutes. Nobody noticed.

[00:38:59] ZB: Precisely, and then you roll it out incrementally from there, right? Yeah, that is what we have seen as far and away the best way to adapt.

[00:39:08] JM: So when Istio came out, I guess two or three years ago. When was the big –

[00:39:15] ZB: 2017. Gluecon in March of 2017.

[00:39:18] JM: Okay. March of 2017. Then was it the KubeCon that summer where they really made their big launch at KubeCon?

[00:39:26] ZB: Yeah, that KubeCon U.S., where it felt like IstioCon almost.

[00:39:31] JM: IstioCon, right.

[00:39:31] ZB: Yeah. That was maybe the peak. I was actually just talking with some people about that exact event maybe 30 minutes ago. Maybe the peak difference between Hype and Istio, the project, versus where the capabilities of the project actually were.

[00:39:44] JM: It was so hilarious, because –

[00:39:46] ZB: It's painful. It's so bad.

[00:39:47] JM: It must have been really painful for you, because something went – What went wrong? Something went wrong with the marketing function. It was like somebody in Google marketing got too much budget for Istio, or like maybe it was the IBM marketing folks actually.

[00:40:01] ZB: Exactly. Yeah, it just took off. Yeah, for whatever reason, at that particular event, there are quite a few contributing factors, Concept Radio at that time that led into this, into kind of the B start KubeCon that year almost, right? Yeah, I think actually that was probably the most detrimental thing that could have happened to the project. Period.

[00:40:22] JM: It was a total banana peel.

[00:40:24] ZB: There's a company that I talk to pretty regularly these days to try and help out with some of their Envoy adaption in particular. I say, "Hey, if you all are using Envoy, you all should really look at Istio."

[00:40:36] JM: They're like, "We've been reading Twitter."

[00:40:38] ZB: Well, no. No. Even worse, they said, "Oh! We already tried it back at 03. I'm never touching that again." Right? So it's like, "But, no. That was three months into the lifecycle of a project. 03 – Istio is six months old then. It's three years old now almost. Actually, it will be three years in two weeks, I think.

[00:40:59] JM: Just real quick to tell the story for people who don't know. Just basically, Istio was given a ton of fanfare, a ton of promotion at this KubeCon, and this was when it was at whatever.

[00:41:12] ZB: I mean, it was like 03, I think actually –

[00:41:14] JM: 03. It didn't work.

[00:41:14] ZB: Yeah. It was super early days.

[00:41:15] JM: It was really hard to deploy, and this was being advertised as the infrastructure solution of the future.

[00:41:20] ZB: The next Kubernetes, right? It's like –

[00:41:24] JM: This is a thing, definitely the thing you want doing your TLS handshakes and all your other security management and your traffic routing and everything, and people, "I can't install this."

[00:41:31] ZB: Yeah, they got so excited, right? Because they were like, "Oh!" You go to the talk and you hear the laundry list of features, the pages and pages and you go, "I need exactly that. I need exactly the observability." As a developer you sit there and you identify with it deeply and try and use it and burn yourself, right?

[00:41:46] JM: The other thing that made it look really bad was that the conference was promoting Istio so much and Linkerd was over there in the shadows, and Linkerd was like people actually uses it in production.

[00:41:57] ZB: Yeah. I know. I have so much respect for William, right? He is on a phenomenal job with Linkerd and with Buoyant. Especially now that I have left Google, I have so much more respect doing the startup game. It's hard, right? So I have a lot of admiration for him having to navigate that, right? He was saddled with this big gorilla that came in. Again, the hype really was painful for him too even though they had a product that works. Because Istio became – Because of the marketing function there, because Istio became synonymous with service mesh in so many people's mind and failure, hard to use –

[00:42:31] JM: It actually ended up really good for him.

[00:42:33] ZB: Well, yeah. Exactly.

[00:42:35] JM: In the long-run.

[00:42:35] ZB: Yeah. But in the short-term there, he was up against it trying to tell, "No. It's Istio. That's hard to use not". That was exactly some of the kind of subtext there in that KubeCon, is people were looking around going, "Well, why are they talking about Istio when Linkerd is the one?"

[00:42:54] JM: But at the same time, it was easy to tell that this thing was going to work out. Even back then, it was like, "Well, okay. We all know Envoy works really well, and we all know that people want this service mesh stuff at some point."

[00:43:11] ZB: Of course, we were in the heady highs of Kubernetes just winning the orchestrator wars, right? Now Google has the next thing.

[00:43:19] JM: What was it like being at – So you were at Google during the container orchestration wars?

[00:43:23] ZB: Yeah. Sure. But I wasn't really working in that side of the house as much. I was in GCP doing enterprisey things there.

[00:43:29] JM: Okay. What was it like seeing that war?

[00:43:33] ZB: I mean, I got my popcorn and watched, because I didn't have horse in the race then, right?

[00:43:36] JM: Same here.

[00:43:38] ZB: I enjoyed it. I thought it was so cool. Yeah! I enjoyed talking – Looking at it just as a software engineer. I'm fortunate that I live in San Francisco, so I have good friends that are

SREs that at a bunch of different places. So I got to hear some pretty first-hand accounts of the war. For me it was always really entertaining to see.

[00:43:58] JM: Did you go to conferences during that time?

[00:44:00] ZB: No. I didn't go to conferences at all either.

[00:44:01] JM: Man! It was hilarious.

[00:44:02] ZB: Oh! I'm sure.

[00:44:02] JM: The funniest part was like walking around the Expo Hall and just seeing the deep confusion. Just like to talk to Mesosphere, they tell you –

[00:44:10] ZB: Like service mesh today.

[00:44:12] JM: Well, actually not really like service mesh today. Today it's like pretty well-defined. It's like you've got Linkerd over here, you got Istio over here, you got a million Istio providers and you got one Linkerd – Well, I guess –

[00:44:23] ZB: There's Mesh traffics –

[00:44:26] JM: Oh, yeah. The new thing.

[00:44:28] ZB: And Kumon. Yeah. There's more entries coming into the field for sure. I think it will be interesting to see if there is confusion or not. I think you're largely right. I think, in particular, the fact that Google is behind it. Coming out of Kubernetes in particular in that timing there with the project did a whole lot with respect to exactly what you just said, which is the general perception that Istio was going to be the one. The other side of it is just look at the money behind it, right? It's Google, IBM. Who else is going to come along and fund the thing?

The only other players in this space that would legitimately be able to fund this competition there are going to be the other cloud providers, right?

[00:45:08] JM: Yeah. I mean, the other difference between the service mesh wars and the container orchestration wars is container orchestration wars, it felt like for as long as there was a container orchestration war, none of the banks and financial companies –

[00:45:22] ZB: Wanted to move.

[00:45:22] JM: They didn't want to move. They're like, "No, we're not going to invest in Mesosphere. We did this with open stack. We made this mistake with open stack."

[00:45:29] ZB: Yup. Again, in the same way that I think some of the halo of Kubernetes extended to Istio with respect to people perceiving that it will be the winner, the same thing happened with decision maker. At least, my perception is the same thing happened with decisions makers looking at the mesh, right? Because a lot of them are literally – Because a lot of these people are the Kubernetes team now, right? They looked at it and they said, "Well, that's what I be on, and now I've got my job here."

[00:45:57] JM: Right.

[00:45:58] ZB: I think that, yeah, that halo effect again of Kubernetes –

[00:46:01] JM: People are not going to shy. People are not deliberating between different [inaudible 00:46:05].

[00:46:05] ZB: No. Not in my experience, right? Certainly, people have looked at different ones. But by and large – Again, part of this is just bias if you're going to come talk to Tetrate. To me, you're probably going to ask us about Istio. Certainly, there's some bias there.

[00:46:20] JM: I'm talking to William later this week, and I am going to ask him like, "How are you going to compete with the Kubernetes community that seems like they're all-in on Istio, and if they're not all-in on Istio, Google will spend money until they are."

[00:46:38] ZB: Yeah.

[00:46:39] JM: How are you going to compete with that?

[00:46:41] ZB: Yeah. This is one of my favorite war games, right? Because I happen to be intimately familiar with this space, and now being at a startup, I have kind of an appreciation there. I think it's a really fascinating question. I am very excited to see what they decide to do and what tactics and strategies they decide to use there. I actually think that there are quite a few different avenues that they could take that would be very successful.

[00:47:03] JM: Like what?

[00:47:05] ZB: In my opinion looking at it, I think that – I don't have super insight into their business, but I look at it and go, "There are two clear ways that you can win right now." One of them is talk about usability. Talk about usability from dawn till dusk. Like, usability, usability, usability. Then the second thing I think –

[00:47:24] JM: Wait. Meaning that Linkerd is more usable than Istio.

[00:47:26] ZB: Exactly. My thing is usable. You can it deployed, right? That's pretty fair. They have put a lot of effort into solving those problems. That is one of the bigger indications I think of maturity in a project, right? Production readiness and maturity are very different things.

[00:47:42] JM: Totally. People may not understand this, but building software that is usable is really hard.

[00:47:46] ZB: Exactly. They've been doing it for years longer than – Right? A lot of the core of Linkerd is original functionality came out of Finagle, right? So like a lot of – It had been vetted and done.

[00:47:58] JM: Yes, iterated on many times.

[00:48:01] ZB: Anyways, so usability I think. Then I think that there still a lot of room actually to target verticals, especially in Kubernetes. I think that you're going to see a proliferation of

vendors that are starting to target verticals and they're starting to – That will start to tailor offerings towards specific [inaudible 00:48:15] industries.

[00:48:16] JM: HIPAA compliance or something.

[00:48:18] ZB: Yeah! That maybe is one. I think maybe the ML stuff is pretty interesting, right? So the Tensorflow and the – So that's a perfect example of a project that's doing a vertical, right? We have machine language and model serving on Kubernetes.

[00:48:33] JM: The Kubeflow stuff.

[00:48:33] ZB: Yeah, Kubeflow. Exactly. Sorry, the name escaped me there. So Kubeflow is a perfect example of a vertical product on a platform, right? We built this product around shipping machine learning. I think that there is a lot of room to do that style product using a mesh, right? That's a place that I will use my educated opinion here and say I don't think that the cloud providers are particularly going to – They're going to move into those spaces last, right? They're going to take their time going for those –

[00:49:06] JM: The verticals.

[00:49:07] ZB: Yeah.

[00:49:07] JM: Well, they got so much other stuff to work on.

[00:49:09] ZB: Exactly. From my outside view of Buoyant, I would love to see them do that. I think that that would be a way that they could be really successful. They even just announced their product today actually.

[00:49:20] JM: I didn't see it went live yet.

[00:49:21] ZB: Yeah. I don't remember what the link is offhand, but you can put it in the show notes.

[00:49:25] JM: What is it? What did it do?

[00:49:26] ZB: They're doing basically management over top of a mesh. So some visibility, some process, that kind of stuff, right? That's a whole – I think that's actually really where we need a lot of tooling built out, right? That's right in the vein of what Tetrate itself is doing, right? So we're really focusing on the management side. We talked about mesh is really a people problem. It's not a technology problem. So management is really – Management planes and the management of this infrastructure is really where the technology side meets the people side.

[00:49:55] JM: Okay. This is the idea that whatever service mesh you're doing, you're using, from a performance perspective, from a deployment perspective, Linkerd will figure it out. Istio will figure it out. This stuff will get figured out. Where the real battle is going to take place is the developer experience at the control plane.

[00:50:14] ZB: Exactly, right? And how can you actually make that tractable, right? Because again, like we said, no business – My customers don't care about the fact that I moved from monolith to a bunch of microservices. Customers care about the fact that they get to see features faster. The way that I do that is I make it easier for my developers to build and deploy software that my users can touch.

[00:50:36] JM: So we know that developer experience matters a lot, but you look at AWS and the developer experience is – I think Google cloud, the developer experience is significantly better than the AWS developer experience. But of course AWS has just more traction, more footprint and it's more well-developed. Maybe it's more sturdy.

[00:51:00] ZB: AWS has a phenomenal job of meeting customers where they're at and giving them things that are familiar to them already. It is a lot easier to be a network admin with all my CISCO certifications and go do my same job in AWS, because I still need to configure VPCs, because we still need to do some of those things, than it is to come to Google where Google says, "No. It's good. The network is flat. You don't need those things. We do it different way."

So this is actually I think some of the fundamental tension, right? It's a fundamental strategy choice between those two cloud providers, right? We talked earlier, Amazon is fundamentally

customer-focused. That manifests as not really being the player that's driving forward the bleeding edge. Amazon is not known for their bleeding edge innovation. But what they do do better than any other company in the entire world is commoditize software, right? They take software that has been proven, that has been built out and they bring it to the masses better than any other company in the entire world, I think, right?

Google does a fundamentally different strategy, right? They say we want – It's because they have a fundamentally different view because their customer is the internal engineer, first and foremost, and not the external customer. They don't care about needing an external network admin where they're at at the knowledge that they have today, because that's not the alternative they have and that's not what their customer needs. So you wind up part of where this manifest is in the U.S. and in this set of primitives that you deal with in a platform, right?

[00:52:33] JM: Yeah. Google Cloud UX is great in my experience.

[00:52:36] ZB: Yeah, me too. Now I no longer work for VCB, and I've had to be a client of both and I can say for sure I much prefer the [inaudible 00:52:45] GCP, right? It blew my mind that AWS, I couldn't have one console. Well, I didn't have one console to see all my stuff. Why is the console spoke to a region? What? There were a bunch of things I look at and I'd go, "Wow! That's kind of weird."

I think this difference in approach in strategy is pretty fundamental, and I actually think that you can look at – And there are some interesting parallels there bringing us back to a service mesh discussion between like Linkerd strategy versus Istio strategy, for example. Istio is very much a manifestation of how Google likes to do engineering, right? It's complicated, but it's really powerful. There are a lot of sharp edges to cut yourself with, right?

Linkerd is much easier to use, right? But maybe it doesn't scratch all the itches that you need. Maybe it doesn't cover all the cases. It's less feature full in some capacities, right? It certainly is not going to do things outside of Kubernetes for you, for example, where Envoy was built to run on EC2 originally. Lyft ran – Right. The original incarnation of Envoy is as edge ingress proxy on EC2 VMs to handle Lyft dropping, right?

[00:53:48] JM: So that's right. So like if I have some Pivotal Cloud Foundry installation that's all on VMs, I could adapt Istio. I could have Enjoy proxies on my –

[00:53:59] ZB: Yeah. So if we're doing like PCF, there are maybe some extra caveats there. That's because of the environment that Cloud Foundry sets up. But like Cloud Foundry actually is an interesting example, because they've been early Istio contributors from the beginning, specifically with an eye towards using Istio in the implementation of Cloud Foundry to do some of the routing and networking things that end users see.

[00:54:23] JM: Okay. Well, we don't have to go down that rabbit hole. So when you were at Google Cloud, you were straight up engineer.

[00:54:31] ZB: Yeah.

[00:54:32] JM: It seems like you've really enjoyed the shift towards more of a go-to-market kind of strategist and sales engineer. I mean, you have to wear a lot of hats.

[00:54:41] ZB: Yeah. A small company wear a lot of hats, right? Definitely what makes me happy is sitting down and getting to do some hard design and ideally writing some node. But yeah, I enjoy the other aspects of it, right? I enjoy getting up and talking with customers and talking with users and hearing about the problems that people have. I enjoy some of these talking like we're doing now. It's always kind of run, right? It's an interesting and different thing, right? Yeah, a small company wear a bunch of hats. That's just kind of how it goes.

[00:55:11] JM: What's the hardest part of building an infrastructure company?

[00:55:13] ZB: I mean, I can speak for us I think. So we made some pretty interesting decisions with respect to how we build Tetrate, the company, right? I'm sure you've talked a little about this with Varun in past episodes. But this is typically how I talk about it. When we talk about tech startups, every company needs to take risks.

The whole premise of a startup is that you take a risk. That's how you make money. Most technology startups, the risk that they take is the technology that they're picking. I feel like we're

pretty secure there. I don't feel like it's very risky. Now, obviously, it's a little self-serving. But I look at Envoy. I think it's pretty rock solid, right? I look at Istio and I'm pretty happy with where we're at.

There are things that need to be fixed, but from the perspective of I need to enable large scale enterprise customers to use this stuff, Istio is in not a bad spot today, right? I feel good about our technology picks. I think the real risky pick that we have is a company, and so the hardest thing for us with respect to this build out and this thing is the fact that we're totally remote. So we're globally distributed, right? I have 27 people in 11 countries and in 10 different time zones. So maybe not quite the answer you're looking for with respect to infrastructure.

[00:56:22] JM: Tell me more.

[00:56:23] ZB: But that's a fundamentally different thing. I actually firmly believe that that will be how basically all companies work in the future. It really is awesome for sure. But it's so hard, because so much of human interaction is predicated on body language and just being [inaudible 00:56:41].

We're recording this podcast in the same room together, because a podcast recorded where we can look at each other while we talk is way better than one where we're recording it over a phone and not just because the recording equipment is better, but because the whole conversation is better because we can see each other, because we can interact with each other in a more tangible way, and humans are just built to do that, right?

So one of the key challenges for a totally remote company then is how do you start to enable that? How do you enable intentional interaction between people? There's no water cooler to go chat about the weekend, right? Because you're in India and I'm here. So if we want to chat, if we just kind of want to shoot the shit about our lives, we have to intentionally take time to do that, right? That manifests in a bunch of different ways. So in general, communication has to be incredibly intentional. That's a hard and different challenge I think than any other companies, right?

So then on top of that, there's just the engineering side – There are plenty of other things that are hard about an infrastructure company, right? The engineering part is challenging. The problem space itself tends to be a little bit deeper and more technical than many other problem spaces, right? There are all these knock on things. But at least from – That any infrastructure company has to deal with, at least from our perspective, the thing that makes it hardest but is also I think maybe one of our single biggest benefits as a company full stop is that we're totally remote.

[SPONSOR MESSAGE]

[00:58:14] JM: If your product has dashboards and reports, you know the importance of making those analytics products beautiful. Logi Analytics gives you embedded analytics and rich visualizations. You don't need to be a designer to get great analytics in your product.

According to the Gartner Analyst Firm, the look and feel of embedded analytics has a direct impact on how end-users perceive your application. Go to logianalytics.com/sedaily to access 17 easy changes that will transform your dashboards. That's L-O-G-Ianalytics.com/sedaily. Logi Analytics is a leading development platform for embedded dashboards and reports, and Logi gives you complete control to create your own analytics experience.

Logi Analytics has been a sponsor of Software Engineering Daily for a while and we're very happy to have them. So thanks to Logi Analytics, and go to L-O-G-Ianalytics.com/sedaily to find 17 easy changes that will transform your dashboards. You can get better dashboards and reports inside your product with embedded analytics from Logi Analytics.

[INTERVIEW CONTINUED]

[00:59:45] JM: What do you think is going to happen to all of Google's napping pods and lunch buffets and like bouncy castles and like buildings all over the world as people realize all of these perks don't add up to the level of happiness I get from sitting at my –

[01:00:09] ZB: Exactly.

[01:00:08] JM: In front of my own computer.

[01:00:11] ZB: Yeah. So I can actually speak to this in a very real way, because I had all those and now I sit at home and I joked with people, but I wasn't really joking. The thing that I miss the most is the food, man. Really good food. So there're definitely pluses and minuses to both, right? I think that it's not really for everyone to sit at home and work at home all day. I think a lot of people have a hard time doing that.

We certainly had some people that have had a hard time doing the transition to totally remote. In general, we hire only open source developers or primarily open source developers, because they're used to working remotely. It can be a tough change to navigate, right? But I do think that the freedom is massive, right?

I love that I can go – If I don't have a meeting in the middle of the day, it's super easy for me to go see a movie, and I have a theater to myself, or more likely I – Yeah, but I'm a big biker. So I love biking around San Francisco, right? I'll go do a ton of midday bike ride.

[01:01:07] JM: Yeah. I go for a run in the middle of a day.

[01:01:09] ZB: Exactly. The quality of life –

[01:01:11] JM: It's so much better.

[01:01:12] ZB: Exactly. I think that the steady state that we're going to land in is that there's going to be a mix of – And we have some office space, because some days you just got to get a pass, right? Some people don't work effectively from home. So you need a space. I think it definitely does devalue some of those perks, for sure. But the other side of that is the number of companies that offer those crazy perks like that is basically like 5. It's the [inaudible 01:01:37] companies, right? There's not outside of that pool. It's not –

[01:01:41] JM: I don't know, man. San Francisco.

[01:01:43] ZB: Sure. But most startups are doing like – Right. Google is like all three meals a day for free. It's like egregiously over the top, right? Certainly, there are. It definitely does devalue at some. One of the interesting perks that you can have as a company. As a company, one of the biggest expenses that you have especially if you're in San Francisco, for example, is your office space, right? We don't spend money on office space. So it makes it much more feasible to start to do things to address some of those perks if we want to, right?

We haven't really needed to. But like we can still easily do that and our costs are so substantially lower than if we were all in-person just because we're not paying for the privilege of having a space, having a door that we can lock, that we can share.

[01:02:31] JM: Now, much of what Google got out of that in-person feeling, and I worked at Amazon for a while, and a bunch of what Amazon gets out of this in-person stuff is this cultural cohesion.

[01:02:47] ZB: Yeah, that's a big part of it for sure.

[01:02:49] JM: Like the spreading of values, and some values which I'm not sure could permeate and exist strongly in a remote.

[01:02:59] ZB: Yeah, and that's definitely part of the challenge, is how do you successfully – What is company culture other than some of the interactions that you have together? So when those are disparate and you don't have the perk of being in the same office together is that we get to see many of those other interactions that don't involve us. That is what informs our own – How we act, right? Because that is the norm. That is what is accepted in this space.

Yeah, that's another place where, again, you have to be very intentional as a remote company. Intentionality is a really, really critical concept I think just across the board. You have to have intentional communication. Part of that intentional communication is consciously enforcing the values in interactions. You have to do these things more and more cognitively. They have to be more in the front of your mind than in a traditional environment, because you have to take those rare opportunities to drive home that this is how we do it.

Then the other side of it is just communicating more. So part of it is writing down norms and expectations, right? that's a good thing for a company anyway to have to start. As long as those are living documents that can breathe and change as the company grows. That's a big thing. So we have a mix, right? So it's a mix of document more make available communications more with each other and be intentional about how we set culture, right?

I think about this, because I was one of the first engineers. I'm in more of a leadership position. So I think about this constantly with respect to how I present myself in meetings. Any leader does to some degree. You have to. But again, with a remote, you have to be even more cognizant of it. You have to be even more – You have to be mindful, right? Intentional. Intentionality, mindfulness, however you want to say it. That I think really does boil down to being one of the critical features, and that's challenging on a team. We are not humans or tend to be pretty selfish by nature. We don't tend to be very good at mindfulness and that. So it has to be a practice, right?

[01:05:17] JM: Right, totally.

[01:05:18] ZB: I would argue maybe that the set of values that you have to have in a remote company, maybe you wind up being a little different than the set of values that you have in an in-person company, especially because of things like that. Mindfulness and intentionality is a very important value that does not tend to appear in many traditional companies sets of values, right?

[01:05:40] JM: Lots of stuff we could continue to explore there. I just want to wrap up, long-ranging question. How do GCP, AWS and Azure in the limit strategically differ from one another?

[01:05:53] ZB: Yeah. I think that's really interesting. We talked about this a little bit, right? Amazon continues to do what they do best, which is we're going to listen to what our customers say. When a sufficient number of them say that they want this button, we're going to add the button. I think they're going to continue with that march. I don't really – Unless some organizationally scaring event happens there or there are some large change in leadership, I don't think that that would change. From their position, why would it? They're the dominant player right now, right?

Like we talked about before, Google is fundamentally a technology-oriented company, right? If Amazon is fundamentally a customer-oriented company, Google is fundamentally a technology-oriented company, right? So they're going to continue to have really awesome tech. I would argue that App Engine today is probably still ahead of its time, and App Engine was released in 2008, right? Obviously, it's changed in that intervening 11 years, but it was a decade – It's arguably 5 – Maybe it was ahead of time and it's a decade old.

[01:06:51] JM: It's hilarious how characteristic that is of Google Cloud's presence in the market.

[01:06:56] ZB: Exactly.

[01:06:57] JM: It's like this is the future. Google Cloud is the future. So it's additionally funny when you see –

[01:07:03] ZB: If you can walk customers there, because again that's the thing, is that Amazon meets them where they're at. So it's comfy. Google doesn't. That is the – If Google can figure out how to bridge that gap and they're trying with some of the – They're trying to do that from an organizational perspective of like [inaudible 01:07:20] and with the growth of the sales side of –

[01:07:24] JM: Well, Firebase, I thought was interesting example.

[01:07:27] ZB: Yeah, Firebase is a good example of that.

[01:07:28] JM: People love Firebase, and Firebase is a new – It feels like a newish technology. It feels like a fresh technology, but it's exactly what the hipster developers want.

[01:07:37] ZB: Yes. But critically it's an acquisition.

[01:07:40] JM: Right.

[01:07:41] ZB: So Google took that technology and they've built it and they've made it better and they've made it cool and –

[01:07:47] JM: They have these cross-sells into the Google ecosystem from Firebase. So maybe they'll do the same with Looker and whatever.

[01:07:53] ZB: Yeah. You can probably rest assure that that will continue to do this strategy, right? They want that interdependence between things.

[01:07:59] JM: It's kind of an interesting strategy.

[01:08:00] ZB: It's a natural. Yeah. So they're going to run that down, right? I think that they will continue to get more traction. There're a lot of small things that manifest there. One of the really nice things about using Google APIs is that they tend to be pretty consistent. I know somebody did an analysis of like the AWS APIs, and in particular they were looking at –

[01:08:19] JM: All million of them.

[01:08:20] ZB: Exactly. They were looking – Now, to be fair, Google has hundreds of that. I was on the API survey. I was on the API team here. So Google has hundreds of APIs too. They're public too, and many less than AWS obviously. But somebody did analysis of the AWS next page token and how do you paginate through a list of APIs in AWS, right? There's a table and it's like the capitalization is different. How many of them have a different field between the return and the next one? So it's not next page token into page token. It's like this mismatch. Google APIs don't have that.

[01:08:56] JM: They're consistent.

[01:08:56] ZB: They're consistent. The reason they're consistent is because there's an internal process in place to do these reviews that mandates that consistency. That is a phenomenal feature that nobody really talks about, but that usability goes through. So I think that in the long-run what will happen is that as developers, where they're at, moves closer to the things that Google is shipping, it will start to make more sense and they'll start to get more traction, right?

Of course, we're also very, very early into the cloud race and that we're not 10% in yet, right? I think that will work out well for Google continuing forward, right? Then of course there's Azure. What does Azure do? They know enterprise sales more than any other thing. I haven't used their cloud. They had to use their cloud. That's not quite the right connotation.

But either of the other two, they understand the enterprise sales motion and they already have their foot in place particularly in the middle of the country.

[01:09:57] JM: Okay. Right. The middle of the country. That's the key, and probably in a lot of like international places too.

[01:10:04] ZB: But there are so many people that will never touch Amazon, but Microsoft, they're already using Microsoft. So they're already like – They're already in the door.

[01:10:12] JM: They'll never touch Amazon, because –

[01:10:14] ZB: Well, for a bunch of different – Right. So maybe they take their – Maybe they think they're competing against. That side of it or for whatever reason.

[01:10:23] JM: Oh! Hilarious.

[01:10:25] ZB: There are a variety of reasons.

[01:10:25] JM: I think Walmart has like a deep partnership with Azure for that very reason.

[01:10:30] ZB: Yeah. So Walmart was the most famous instance of this, because they tried to make their vendors not do it. So no, I think that Microsoft will continue to rundown that enterprise sales side. They're very good at that. I think that concurrently they will continue to bulk up their operational side and build out, right? So it's kind of funny, because Microsoft in some ways – I worked in a .NET shop for years. In another life, I would have loved to have gone to Cambridge, gotten a PhD in type theory and had gone and worked for Microsoft Research.

[01:11:01] JM: Right. Go hang out with Anders Hejlsberg.

[01:11:02] ZB: Yeah. Exactly. Right. Anders was one of my heroes, right? As a younger engineer reading about – So in so many different ways, Microsoft does really cutting-edge technologies and really visionary things. In a lot of ways, the do Microsoft Office and Outlook and these institutional things that have the weight of an institutional enterprise thing and they're not new and fast. But they get the job done.

I think their challenge is going to be how do we kind of bridge these two worlds together, because that's where I think they get the really compelling things that neither Google nor AWS will do. So their challenge is going to be – So they have the sales motion. Like if I'm going to build the dream cloud, I would take the enterprise side of Microsoft and I would combine it with the technology side of Google and we would go conquer the world, right?

[01:11:56] JM: Well, what about AWS? What happens to AWS? You'll just like throw it out? Throw it all out.

[01:11:59] ZB: No, they're phenomenal, but it's not the – I don't think that fundamentally they're –

[01:12:03] JM: You can use AWS Lambda to stitch your two sides together.

[01:12:07] ZB: Yes. I don't mean to disparage AWS or anything, but if there are realist and idealists on the engineering perspective, I'm much more an idealist, right? I want the world to move into the better UX. I think that it's absurd that I have to like peer VPCs. This is 2019. I want these two things to talk. I told you I want these two things to talk. Why do I have to – Why do I have to do –

[01:12:34] JM: So you're not enamored of these serverless AWS – I think the serverless stuff is pretty futuristic. So that was totally novel.

[01:12:40] ZB: Yeah. Serverless I think is really, really fascinating. I think that's it's a really immature architectural pattern today and that we need – The transition from monoliths to

microservices is already incredibly hard in large part because tooling does not exist to cope with that, and now you're going to do even more? You're going to take it to an even bigger extreme?

[01:12:59] JM: No. But it's something different. The serverless paradigm –

[01:13:01] ZB: It isn't. So I look at serverless as being largely two categories of thing that are conflated or that are frequently conflated together. One of the categories of thing is I want to write only business logic. That's been a goal of software engineering since we invented programming language, right? We invented the first one and then we said, "Oh, this is real bad. I want to write," right?

Whether or not we ever get to that goal, I don't think any particular architectural or deployment paradigm is going to solve that problem. Then there's the other half of what serverless platforms are today, which is the operational side. I want it to have logging and monitoring and alerting out of the box. I want it to have the visibility. I want to have scaling to zero and automatically scaling. These are all things – Ideally, I want to be able to scale up and down and hopefully I could scale to zero, because that would be really nice if I could.

So if you look at those two sets of things – So let's discount the business logic only. I don't think that that's a realistic goal in the near term. Then we start to look at that platform that is serverless, right? A large majority of the features that are there look very similar to the mesh features. So then in my mind, it comes down to, "Okay. So then what is the – So what are the differences here?" And they boil down to a couple of different things.

This focus on small deployment units, and in particular, the requirement that basically you rewrite code. Any new architectural pattern that says, "Hey, if you want to use me, you need to rewrite everything that you have," is a non-starter.

[01:14:40] JM: What are you talking about? Like we're writing in Lambda sense?

[01:14:42] ZB: Yeah. I can't take my monolith and split my monolith into Lambdas. It doesn't decompose that way, right? Instead, you get told, "No. Go re-implement it," right? But I mean thanks we're still running COBOL mainframes. They're not going to re-implement everything to

go to Lambda. So you need something that gives the feature set of that platform, right? You want the auto-scaling. You want the operational things out of the box. I don't think that that should be coupled with the deployment units that is that small.

[01:15:15] JM: I mean, I appreciate what you just said. I think what is interesting about the Lambda model is it's basically like, "Look, if you're all in on AWS, here is a very smooth interconnected developer experience." You get DynamoDB, you get Amazon Elastic this and Elastic that and it's all connected with Lambda functions. It all works fairly smoothly. Yes, you have to interface it.

[01:15:39] ZB: So that was the App Engine too, right? App Engine was this whole – If you write Python. It was. It was this walled garden.

[01:15:46] JM: It was super walled garden, but it was also like just not as expansive as the AWS version. It's vision is like we give you everything in the kitchen sink and tons of other stuff and like it's just super interconnected and it's super diverse, and Google App Engine back in the day was like, "Yeah, just a Python application," which was fine for a lot of people. That was more of a marketing issue, but AWS has good marketing so people can actually adapt this – Anyway, I mean –

[01:16:07] ZB: No. I think it's – Yeah. No. I think it's really interesting.

[01:16:16] JM: Reinvent is next week. It's not Reinvent. I forgot. Oh God! I thought it was at Reinvent.

[01:16:21] ZB: Yeah. Look, Lambda and engine roll, this serverless idea, is really compelling, right? The idea that we need to be able to –

[01:16:31] JM: It's super proprietary, super blatantly like we're locking you in and you are never getting out. You're going to love it.

[01:16:37] ZB: Yeah. I just think that – And then that aside, just the mechanics of it today, the units about it are wrong. I do a lot of API design. That's part of what I did in Istio, and a lot of

modeling. I like type theory, like I said, right? Unit of edit. What is the thing that I have to change and that I have ownership over is one of the things that we think about a lot with respect to API design.

Even vetted in Lambda as it is today just isn't right and it doesn't conform with the tooling and it doesn't really conform with deployments. So I totally agree that you want this heavily interconnected thing. That's the nirvana that we want to get to, right? It's all the cool stuff that I need to in my app just works right there, and I don't really have to think about it. It's that idea of I get to fill this on the business logic.

[01:17:20] JM: That's what it's like working at Google, right? You just import a library and it all works magically. No? Okay.

[01:17:25] ZB: Yeah, exactly.

[01:17:26] JM: Okay. Maybe not.

[01:17:28] ZB: Jeff Dean walks by and he types on your keyboard and then everything works.

[01:17:31] JM: Yeah. He's like, "Oh! You forgot this import. You forgot to import the magic thing.

[01:17:35] ZB: Yeah. That's one of the things that is most disappointing maybe to people when they joined Google, is that it turns out the sausage – Like making sausage is a nasty business, and it's messy everywhere, right? I don't know. There definitely are – Google has done a good job of handling a lot of the problems that serverless tries to – There are like internal projects that are how you would build software as a developer Google today internally.

I think those models that they're using, which you can extrapolate from App Engine, if App Engine was developed in 2008. Now, jump forward 11 years. If they've been developing at that pace for the same time.

Those models centers on write the unit that you've been writing. Write a service. It should be small. It should be focused on the business logic. We have a lot of those libraries that handle a

lot of the other things that have been build up overtime, right? But you're going to write – Here, you have an object. You own that object. This service owns that object. Write all the stuff and put it together. Then we'll worry about how it actually gets deployed. We'll look at the dependencies that you have. We'll look at what it's using. We'll look at the things that it's consuming and stuff like that and we'll figure out where we actually want to do some of the deployments for how we want to assemble the binary, things like that. You don't need to think about it.

[01:18:48] JM: Yeah.

[01:18:50] ZB: That kind of a model is much more compelling, because the tooling that I am used to dealing with is the same. How do I debug this binary? Well, I just run the binary and I attach my debugger to it. How do you debug a distributed Lambda application on your box? This was always the problem that App Engine had. App Engine didn't have local debug. They had a local debug environment, but it didn't match the App Engine environment. So one of the most painful things that heard from users continuously was this, "Well, it works in your App Engine dev that I ran on my local box, it works. But then I deploy it to the server and it doesn't work." This is a fundamentally hard problem.

[01:19:29] JM: Remote debugging.

[01:19:31] ZB: Well, not even just remote debugging, but how do you provide this environment? Part of the appeal of Lambda is that it's this highly interconnected environment with all these services plugged in. How do I ever debug that? What's my developer – What's my debug workflow? I'm used to running in process in my terminal and attaching GDB. How do I debug it?

But if you with the more traditional deployment model but you handle a lot of the operational pain, then suddenly you can start to unify this development experience and you can start to use the tooling that you're used to, but you can get a lot of the benefits that you may – A lot of the platform benefits of serverless, right?

If I were going to revise, then my two split between – Into three, which is the third one being, to your point, kind of that interconnected glue side of it, right? The dream platform.

[01:20:21] JM: Cloud Nirvana.

[01:20:22] ZB: Exactly. Then, no. Service mesh wouldn't address that, for example. I think that is a desirable thing. I don't think that that is necessarily predicated on the current incarnation of serverless.

[01:20:35] JM: Zach, great talking to you. Great conversation. We went over, but great –

[01:20:39] ZB: Yeah. It will be a nice incendiary thing to end on. I'm looking forward to all the serverless people.

[01:20:44] JM: Wonderful. Thanks for coming on the show.

[01:20:46] ZB: Thank you. Thanks for having me.

[END OF INTERVIEW]

[01:20:57] JM: Email has been around for longer than I've been alive, but there's been surprisingly little innovation in inbox management. SaneBox is a new way of looking at your inbox that puts features like snoozing, and one-click unsubscribe, and follow-up reminders as first-class citizens.

If you are overwhelmed by your inbox and you're almost ready to declare email bankruptcy, tryout SaneBox. In the onboarding process, SaneBox analyzes your emails and helps you sort them into categories. You can get a free 14-day trial and a $25 credit by going to sanebox.com/sed. That's S-A-N-E-B-O-X.com/sed.

These days, I spend more time in my inbox than I do in front of my coding environment, and back when I was programming a lot I would spend hours configuring my coding environment because I wanted to maximize productivity. If you spend as much time managing email as I do, it's crazy not to set yourself up for success with your inbox. Stop the craziness, get sane with SaneBox. Go to sanebox.com/sed and get a free 14-day trial as well as a $25 credit.

Thank you to SaneBox for being a sponsor of Software Engineering Daily.

[END]