EPISODE 08

[INTRODUCTION]

[0:00:00.3] JM: The word “microservices” started getting used after a series of events. Companies were moving to cloud virtual machines. Those virtual machines got broken up into containers, and the containers fit to the size of the service. Services that are more narrowly defined take up smaller containers and could be packed more densely into the virtual machines, hence the term microservices.

As this changed to software architecture has occurred, the DevOps movement has encouraged organizations to have better relationships between development and operations. Continuous deployment leads to fewer painful outages. Improved monitoring tools make it easier for developers to take on some of the pain that was previously centralized in operations.

Several months ago, I attended the Microservices Practitioners Summit, which brings together engineers who are working with microservices at their companies. The conference was organized by Austin Gunter and Richard Li of Datawire. In this episode, they joined me for a conversation about microservices and the summit.

Software Engineering Daily is having our third meet up; Wednesday, May 3rd at Galvanize in San Francisco. The theme of this meet up is *Fraud and Risk in Software.* We’re going to talk about ad fraud, we’re going to talk about fraud that occurs at Coinbase, and we’re going to have great food, engaging speakers, and a friendly, intellectual atmosphere. To find out more, go to softwareengineeringdaily.com/meet up.

[SPONSOR MESSAGE]

[0:01:39.3] JM: Flip the traditional job search and let Indeed Prime work for you while you’re busy with other engineering work, or coding your side project. Upload your resume and, in one click, gain immediate exposure to companies like Facebook, Uber, and Dropbox. Interested employers will reach out to you within one week with salary, position, and equity upfront. Don’t let applying for jobs become a full-time job itself. With Indeed Prime, jobs come to you.

[INTERVIEW]

[0:03:09.5] JM: Austin Gunter and Richard Li are organizers of the Microservices Practioner Summit. Austin and Richard, welcome to Software Engineering Daily.

[0:03:17.6] RL: Thank you.

[0:03:18.1] AG: Thanks for having us.

[0:03:19.3] JM: We've probably done 50 shows relating to microservices, and in this episode I'd like to take a broader view and understand some of the historical context especially since you guys organized a conference around microservices. People started using the word microservices after a series of events. I want to hear both of your opinions as to whether this is the correct series of events that led to microservices.

From my perspective, companies were moving to the cloud. It was virtual machines. These VMs got broken up into containers and the containers can fit to whatever the size of the service is, and that led to the shrinking of programs, because you could more closely fit the memory footprint of your service to the size of the container, and that led to the idea of microservices.

Is that an accurate historical retrospective to how we got to microservices?

[0:04:20.6] RL: Jeff, I think that's an accurate perspective around the technology trend that got us to microservices. The other trend I would point to is also the organizational trend. What I mean by that is that lots of cloud-native companies like Amazon, or Netflix, they were running to the problem where their application was getting more and more complex and they needed to write more and more features. To do that, you hire more and more engineers, but then the question becomes how do you actually ensure your 500th engineer is as productive as the first engineer?

We all know that that's actually pretty hard to do, and so organizationally, they face this challenge of scale and that's when they started thinking about how do we take this big, huge thing and breaking it into smaller chunks, and at the same time that technology trend that you mentioned was happening. They were able to actually capitalize on some technology innovations that weren't possible when they started building their cloud application. When Amazon launched, there was no such thing as a virtual machine or, let alone, containers. It was a combination, I'd say, of organizational scale challenges as well as technology evolution, if that makes sense.

[0:05:38.2] JM: What did the companies look like before this microservices trend got popularized?

[0:05:44.8] RL: I think it was a very traditional development, a very traditional STLC where they would have teams of engineers that would work on different branches, but all on one big codebase. That codebase would get released on a predictable cadence, whether it is weekly, monthly, quarterly. There was a team of release engineers whose job it was to integrate all the different features into "the stable release".

[0:06:13.7] JM: There are some companies that proudly remain monolithic, they have a monorepo and they argue that this a more productive way of doing development. What's the tradeoff there, and what are those companies that advocate the monorepo approach?

[0:06:33.1] RL: First of all, I think you can do microservices successfully even with a monorepo, and Twitter is a great example, of Google is a great example where they have a monorepo. The key thing with microservices is that each team is able to release independently. With Twitter,

every time there's a new feature, that team that finishes that feature doesn't need to check with the other teams. Same with Google or any other company that's adopting microservices.

I would say that the benefit though of a traditional monolith where you have a single application and a single codebase is that it is simpler. If you have a small team of 3, 4, 5 people, it really doesn't make sense to break your application into lots of little services, because at the end of the day you have a small development team. You don't really have a problem with organizational scale. It's only as you start to grow and you have 10, 15, 20, 25 engineers. For every company, that inflection point is different.

It's only when you hit that inflection point where your engineers are starting to run into problems where they need to coordinate with another engineer to ship a feature, or architecturally running into some limitations around languages and platforms. That's when microservices, I think, become much more compelling.

[0:07:53.7] AG: Jeff, one of the funny anecdotes is DHH, who has really famously come out against a lot of things about the size and the shape of tech companies and he's come out against VC funding and he's also come out against microservices as sort of an outspoken critique of that. The example here is that he's running with Basecamp, a fairly small company that's not going to be running into those issues. His point about the magnificent monolith in that Medium article that he wrote, it works great for what they're trying to do. It's just not the right place to think about scaling with microservices.

[0:08:34.6] JM: Austin, you put on the Microservices Summit recently. You were largely responsible for the organization and the selection of the speakers. How are you choosing the presenters for the Microservices Summit? By the way, these are great talks. I sat through several and I watched several online. They are all posted online, and they were really — Microservices Practitioner Summer, I'm sorry. They were very practical presentations for people who are actually working with microservices. How did you select the people who presented?

[0:09:09.6] AG: It's a really great question. Yeah, people can watch all the videos on microservices.com. They are all available there and on YouTube. I think the questions that you were asking about what are microservices? When do they make sense? What's the difference

between microservices and a monorepo. The answers that Richard gave you, I think, illustrate the need for people who have been there and done that, and that's why highlighting the word practitioner is really important.

We saw a need in the industry for a place where engineers who had actually done the work scale their companies, re-architect it from microservices, distribute a development, distribute its systems perspective. We saw the need for those people to be able to come together and share what they had learned with the wider community.

Because it's one thing to say, "Hey, you should break apart your monolith because of microservices." That leaves it in the buzz world realm. We were very focused on finding speakers who had a lot of firsthand experience getting their hands dirty, solving these technical challenges.

Finding the speakers was a team effort. There were ideas that came from a lot of different places about who had done really, really interesting work. Matt Klein was a really great example because of what he — Coming from Twitter, and then working on their microservices stack and then saying, "I want to change some of the things that Twitter had done."

When he joined Lyft and working on Lyft's Envoy, he's got an amazing wealth of just firsthand knowledge that he can offer engineers who are trying to solve similar problems. His talk was fantastic.

We also had Susan Fowler who was at Uber and is now at Stripe, and wrote a book, wrote an O'Reilly book about microservices and standardizing your architecture. When I talked to her about potentially speaking, I said, "What would you do over again if you're having to start from scratch? How would you set up a team that started from scratch around architecting for microservices? How would you set them up for success?" Those were some of the things that she talked about.

My philosophy, my overarching philosophy in finding speakers was find really, really talented engineers, look for as much diversity as you possibly can, and then all I did was have a conversation with these folks and say, "What are some problems that you saw that you don't

see people talking about enough, or what problems are you currently solving that you're excited about that you want to talk about?"

When you're bringing smart people who are doing interesting work together, it's really easy to get them to want to share what they're working on without having to guide that too much.

[SPONSOR MESSAGE]

[0:12:25.3] JM: For years, when I started building a new app, I would use MongoDB. Now, I use MongoDB Atlas. MongoDB Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running.

MongoDB Atlas is the only database as a service from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous back-up, VPC peering, monitoring, and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy-to-use service. You could forget about needing to patch your Mongo instances and keep it up-to-date, because Atlas automatically updates its version.

Check you mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free. Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature.

Get started with a free three-node replica set, no credit card is required. As an inclusive offer for Software Engineering Daily listeners, use code "sedaily" for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out. Thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW CONTINUED]

[0:14:26.1] JM: In watching these talks, I saw plenty of similarities between the different presenters and the different situations that people were in. What about the differences? What were the conflicts in what different people said, or where were the situations where somebody said, "You know, I understand what that previous said, but I completely disagree with them." What were the fundamental disagreements about things that are going on in the microservices ecosystem?

[0:14:57.4] RL: I think there's a bunch of different ones, and I think a lot of them are basically boil down to different organizational challenges. One of my favorites was contrasting Nic Benders' talk — Nic is he chief architect at New Relic. With Susan Fowler's talk, where she talked about her experiences at Uber. Uber, when she joined, was going through hyper-growth. They were going from 850 engineers, to 1,700 engineers in six months.

As a consequence, it was really the technology wild-wild-west. Culturally, I think that's a little bit of Uber as well. Her approach and recommendation was a clear set of guardrails to ensure that everyone's aligned to a common architecture and goal.

If you look at Nic Benders' talk, New Relic, high-growth cloud-native company, but not growing as quickly as Uber, certainly not in terms of hiring engineers. What they decided to do was to literally let the engineers organize themselves into their own teams. I think that that approach worked for New Relic. Susan's approach worked for Uber, but you probably don't really mix the two.

[0:16:14.8] JM: Did you have a favorite war story that you saw in the presentations, each of you? Because I thought a lot of these talks followed the pattern of, "Okay. We got into this horrendous situation and here is how we got out of it, and here's the architectural lesson that we learned along the way." Maybe if each of you have a story that you heard at the Practitioner Summit that was interesting to you. I'd love to hear it relayed.

[0:16:40.2] AG: I think Nic's talks was one of my favorites, and the reason it was one of my favorites was because he spent his entire talk talking about the people and cultural aspect, and I think that's a really important underrated part of microservices, because, fundamentally, you're talking decisions that are historically used to be made at a very senior level, like, "When do we

release?" That's a team decision where you have a release manager and your VP of engineering and everything. You're basically saying, "It's no longer a VP level decision. It's really a team level decision." That's a huge organizational shift.

I loved how Nic talked about that kind of decision making and pushing that down to the leaf nodes, if you will. That was just fascinating to see how they were willing to really experiment, try something, they made some mistakes, but they figured out a lot of things along the way. To me, that was definitely one of my favorites. I also know plenty of people who thought it was terrible. Your mileage may vary.

[0:17:41.5] AG: It was really interesting hearing different attendees talk about various talks that were their favorites, or their least favorites, because you can almost tell where they're at in terms of the type of organization that they have and the size, the scale, and the problems that they're solving based on which conversations they thought were — Or which talks they thought were more relevant.

Nic was talking very organizationally about microservices rather than being more technical and talking about things from a code perspective and an architecture perspective. I think one of the things that was interesting was hearing some attendees who really were like Nic's talk. I realized it was because they're still thinking about things from an architectural perspective rather than understanding how organizationally — Or understanding that microservices is also an organizational, people-centric architecture, not just a technical one.

I think there was — Which goes back to the whole thing about, "Are microservices right for your organization?" Yes and no. It really depends on where you're at and what problems you're trying to solve. It doesn't help that the whole thing has gotten a little buzz-wordy that people don't understand there's a lot of nuance in there.

[0:18:56.0] JM: Certainly, there is this organizational shift. There's this technology shift. It's hard to decipher which one is the cause and which one is the effect and how they intermingle with each other. In case there are new technologies that have been enabled or necessitated in this move to microservices, one that comes to mind is the service proxying stuff that Matt Klein works on. Another is distributed tracing technology.

Richard, given that you've been working in this space, developing products for as long as it's been around, what are the technologies that you're looking at right now most closely?

[0:19:43.0] RL: Great question, Jeff. The technologies that we see the most community momentum behind would be Kubernetes for container management and orchestration. Obviously, Docker. Then, in the service proxy space, we see a lot of momentum behind Lyft Envoy. I say that, especially, because we're seeing not just companies adopting Envoy in deploying it for their cloud-native application, but we're also seeing some commercial vendors picking up Envoy, putting real engineering resources into it.

Now, you're starting to see an ecosystem of not just the engineers at Lyft maintaining it, but you're also starting to see contributions from Google, or IBM, or Datawire all pushing patches, writing documentation are pushing this into upstream Envoy. I think the sign of a healthy community is that it's no longer dependent on a single entity to actually sponsor and drive the development forward.

[0:20:44.0] JM: Service proxying — For those who haven't heard the episode that I did with Matt Klein, or who haven't watched the talk on microservices.com. Service proxying is this idea where you have a program running on all of your hosts that adds a layer of enrichment to all of the inter-service communications or it decodes the layer of enrichment from the microservice that sent the message. You get this standardization of messaging between your different microservices.

I worked at Amazon and I remember at Amazon, there were some kind of service proxying layer. I didn't really understand what it was doing at the time. At Amazon, the onboarding process takes six months because you're just figuring out all these tools and technologies. One of the things is, I believe, what you would call a service proxying layer, where you're like, "Okay. Every time I send a message from this server, why is all these stuff added?" It's like, "Oh. It's because that request is going through a service proxy, which is adding things like distributed tracing and whatnot."

You also mentioned Kubernetes, and I think of Kubernetes is also this sort of standardizing layer, because you're putting all of your services in containers. You're using Kubernetes as the mass orchestration tool. Why is it that we need Kubernetes, but also a service proxying tool like Envoy?

[0:22:15.5] RL: Kubernetes. You need Kubernetes so that your containers have some place to run. You need to be able to manage the life cycle of those containers. That's where Kubernetes really shines.

The other question, which is, "How do the containers actually talk to each other?" is a question that Kubernetes doesn't actually address. The way I like to think about it, Kubernetes is, "Where does my container run?" There's a whole bunch of questions and nuance to that, and that's what Kubernetes excels at. Then, there's the question of, "How do my containers talk to each other?"

Talking to each other means; how do I find the container? How do I — If the container doesn't respond to me, how do I retry? All those kinds of things. That the semantics that Envoy or a service proxy provides. Those are some of the essential pieces that you need. The third piece that you need is actually how do you monitor all those stuff. I see things like Influx, or Prometheus gaining quite a bit of traction there.

[0:23:10.6] JM: The other question around Kubernetes I wanted to ask you — There was a lot of talk, I think it was around this time last year, or six months ago or so, where people were starting to say, "Okay. It looks like Docker is — In some sense, it's getting obviated by Kubernetes," where everybody is using Kubernetes because it's the orchestration layer, and you're orchestration Docker containers. Do they really need to be Docker containers? Maybe it could be a different type of container.

This discussion is further complicated by the fact that Docker is a company that raised a ton of money, and I think they're probably — They have plenty of money left over. They could do something different. The obvious direction for them would have been the container orchestration tool, maybe some sort of SAS tool using that container orchestration tool. That container

orchestration tool that they built, which I — What is it? Docker Swarm, I think. Was it? I don't think it's getting much traction.

I think, at this point, Kubernetes is kind of winning the war. You could maybe say the same thing about Mesos. I have plenty of people who say, "Look. You're looking at it wrong. If you look at this, it's a winner-take-all sort of thing." I don't know, maybe you want to talk about that.

I don't know. I'm struggling to think of the right question to ask, but what's your current assessment of the competitive landscape and where does Docker fit in going forward?

[0:24:46.3] RL: I think Docker as a container format is a universally accepted standard. Obviously, there's Rocket, and I was excited to see how both Docker and Rocket are incubated now by the Cloud Native Compute Foundation.

To answer your question, I think Docker is still a very standard format for how you actually deploy software with its dependencies into some sort of runtime, whether it's Kubernetes, or Mesos, or Amazon ECS, or what have you. I don't see that going away anytime soon.

When we talk to clients and customers, when we say, "We depend on Docker." There's really not a lot of pushback. People say, "That's obviously what you should be doing."

I think, in terms of the commercial prospects, I agree with you. It seems that a lot of the community momentum has sort of naturally gone to Kubernetes. I wouldn't discount Docker just because there are a bunch of smart people at Docker. I'm sure they see the trend even better than we do.

I also think that Docker also has a whole bunch of other interesting technologies and they're trying to build sort of a more robust ecosystem around Docker with Docker Hub, and they've got a huge team. Like you said, a lot of money.

I think that there are probably a bunch of other things that they are doing besides Docker Swarm that could lead to sustainable commercial success. Like you said, when you raise a billion dollars, it's a lot of money to spend.

[0:26:23.7] JM: I think this kind of company trial can be quite beneficial in the long-term, because if it works out — If the company works out, which it probably will, they will have a sense of gratitude. It's not like Facebook or Google where it's been up into the right the whole way. This is a stumbling block, and this is the kind of stuff that can make a company a lot stronger in the long-run.

[0:26:51.4] AG: Yeah. That which does not kill you makes you stronger.

[0:26:55.5] JM: Right. Back to the Microservices Summit. I've been doing this podcast for a couple of years at this point and I've gone to a number of conferences at this point and it feels like — I spent a whole of time in this space, but it certainly feels like the number of conferences is going up and up and up and up. Austin, how do you differentiate a conference these days?

[0:27:19.2] AG: I think that, as with everything in a developer community, it really comes down to the rubber meeting the road; did you have the best possible speakers and did you create a really inclusive positive environment where engineers from all sorts of backgrounds and walks of life, different cultures, et cetera, can come together and attack a particular technical challenge of set of technical challenges in a productive way and come away knowing that not only can they justify the cost of travel and time away from work to their manager, but they have new ideas and they're inspired about something that they can bring to what they're actually working on.

Part of the reason that we kept the Microservices Practitioner Summit semi-small was to foster a sense of intimacy, and community, and connection there, because you get to a really, really massive conference, and it's hard to — There's a lot of different talk tracks, and the summit only had a single talk track. That allows people to focus on — They know what talks are coming up next and they're not sitting there like kind of doing the conference crap shoot of, "Which talk do I go to?" "Is this one actually going to be valuable and useful for my time?"

[0:28:54.4] JM: Right. You sit down to lunch and you can talk to people with a shared understanding of what was just viewed.

[0:28:59.8] AG: Right. Yeah, exactly. It fosters better conversations as well.

I think that that's one of the ways that we've done it with the summit is keeping it small. Modern drama is kind of one of our inspirations for that. It's like it's never going to be a huge conference. The purpose is you're coming, you're going to learn about a specific set of things. You're going to be rubbing elbows with peers who are solving similar problems, have solved similar problems, so the quality of conversation can automatically be very, very high. You have a shared context about what you want to talk about, what you came to talk about.

The other thing is that keeping it small that way and by kind of handpicking the speakers based on their background and their experience, we're able to keep the quality of talk very high as well, because I'm more or less — I wasn't approving what people are going to talk about, just for the record. I wanted to find good people that I knew were going to talk about intelligent, interesting things. I think that's how you set that up.

Knowing that I'm bringing all these attendees together, getting people in the room is a challenge as s conference organizer. I believe that on the other end of that, there's a commitment to making sure that when people leave, they're like, "Yeah! Heck yeah! I love that. I want to come back again, and I want to tell some of my peers and my colleagues that this was a really worthwhile use of time." I really think a lot of it has to just come down to the actual experience and fostering word of mouth that way.

[0:30:38.7] JM: I've heard people say recently it's peak conference, there're too many conferences. I don't think so. I think the developer community is expanding more rapidly than the number of conferences. You look at something like Strata. I don't know if either of you have been to something the size of Strata, Strata + Hadoop Conference, but it's almost like you're at the Super Bowl and there's so many people, the Expo Hall is this gigantic several football field size room of vendors, and then you go there in the evening and it's this celebration of food, and drink, and sales, deals being done, and it's totally different than the Microservices Practitioner Summit, albeit not worse. It's just completely different.

Then, if you go to something like Facebook F8, or Google IO, it is this bacchanalia of food, and drink, and luxury, and seduction by the giant corporation. That is yet another different form of the

conference. What's so great about these things is they put you in meet space with other developers, because so much of our interaction, especially with the increased emphasis on remote work. So much of our interaction is disjoint and virtual.

When you go to that meet space and you sit down with people, you have a different experience. You have a shared experience. You can have a really rapid transmission of information.

[0:32:08.4] AG: I think what's funny about you asking this or you're phrasing this is — I'm from Austin. I'm Austin from Austin. Ha! Ha! Thanks mom. Yeah.

[0:32:19.2] JM: By the way, I have four or five friends from Austin named Austin who have the same thing. I don't know — Did your mom literally think it was funny? What's with that? Why is that a trend?

[0:32:32.0] AG: I think it's just Texas pride, to be honest. I think the answer to your question is I actually — Being from Austin, I was going to South by Southwest interactive very, very early, because I was starting — I was working for tech companies, basically, right out of school. I got very inundated with just — Use the word bacchanalia, which I'm going to have to look up later. Thanks, Jeff, for making me feel dumb.

You go to South by Southwest — The original intent of South by Southwest was kind of the original intent of Austin, which was this small removed community of smart, weird people, either doing a lot of nothing, or every once in a while, doing some cool projects together. Some of that is tech, some of that is art. You've got all these great music history that comes out of Austin. South by Southwest interactive started out like a few tiny minds — Sorry. I tiny gathering of a lot of really creative innovative minds.

As always happens, people who are doing cool stuff draw attention to what they're doing whether they intend to or not. South by Southwest has sort of become this mass orgy of advertising agencies who are working with tech companies and Target and the game that you play at South by Southwest is don't pay anything to eat or drink while you're there, and it's a very easy game and everybody wins, because you walk into some booth — I literally walked into this Doritos mashable thing and they were like, "Hey, we've got Cuban sandwiches on a

grill. By the way, how many mojitos would you like before you walk out the door?" It's like, "I want nine Cuban sandwiches and twice as many mojitos. By the way, I'm here for work." "Wait. What?"

After this one year at South by Southwest where I was like — You're out all night and then you're trying to get up in the morning to be productive again, but I didn't find anything that was actually useful for my time there. I wrote a blog post that actually got quoted in the New York Times the next year about, "South by Southwest isn't a tech conference anymore. I think it's a tech festival."

If you think about it more like a festival rather a conference, it changes your perspective on it. That was around the time that XOXO and Yes by Yes Yes. We're coming out to XOXO. I don't think they're going to do it anymore, but it's this really conference in Portland where they just creators come and share their creations. Then, Yes by Yes Yes happens down in Palm Springs in Southern California every year. They just basically take over this hotel for a weekend and a bunch of really smart people come and hangout.

I think that if you look at these big conferences, part of the purpose of them is like, "Hey, we're a big company," like Amazon Reinvent. They are just there to show you a bunch of cool stuff and then facilitate a lot of conversations. It has a different purpose than like the Microservices Practitioners Summit might, because you can just — It's designed to be something where you're solving problems as supposed to being very vendory.

I think when you go to a big conference as an attendee, you have to understand that the organizer has spent the last year of their life putting this together and they're going to facilitate interactions between you and people that you need to talk to as much as they know how to, but you have to take that into your own hands and have a plan for where you're going to go, and who you want talk to, and what you want to get out of it, and you have to be relentless about it, because otherwise the — As a wise podcaster once said, the bacchanalia of food, and drink, and seduction is going to distract you from what you actually came there to do, and you're just going to be exhausted at the end of it and go, "What did I get out of this?"

[0:36:23.3] JM: This, by the way, is totally illustrative of why I left Austin and probably, to some degree, why you left Austin, where tech is a lifestyle. It's a, "Let's go to work and then after 4:30 p.m., let's go chill at Barton Springs and have a mojito.

In San Francisco, it's like, "Let's work our asses of and make that a very separate —" Anyway, I don't want to go out on a tirade here, but I think what you said where it has gone — South by Southwest may have gone — I'm sure South by Southwest is so big that you could take five different people and they have five different experience at South by Southwest these days. The centricity is a festival. It is no longer a, maybe, intellectual-focused, "Hey, let's talk about new technology and how it's going to change the world," sort of thing. It's more like a, "Hey, let's talk about new technology and get wasted."

[0:37:27.4] AG: It's Silicon Valley on spring break in Austin, and that's —

[0:37:30.3] JM: That's right! 24/7, 365.

[0:37:33.1] AG: That's part of what Austin is different than San Francisco. That's a whole other conversation. You're right.

Again, how do you actually provide value for your attendees? I think it really just comes down to, "What do you — Have a purpose?" Again, the Microservices Practitioner Summit, when we're talking about this in Datawire, the idea was how do we take all these questions that we know people are asking, we know people are trying to solve, but they're not solving them out in public. They're not — A lot of these internal knowledge is in the hands of some really brilliant people who are actually smarter than the rest of us. Let's just be honest. Matt Klein, Susan Fowler, these people are just absolutely brilliant.

Susan Fowler used to work at CERN doing particle physics. Nobody is trying to hire me to go work at CERN, but the problems that she's solving and the thing that she's doing, there are engineers all over the place who are trying to wrap their heads around these problems that have already been solved internally at these various companies. How do you actually put that out in the forefront and share that from a community perspective and from the values of open-source perspective. That was really what the summit and Datawire were trying to accomplish there.

[0:38:45.1] RL: What was really awesome was that people actually want to share. I think that was the thing where when we started doing microservices, we just try to talk to every single smart person we could find. Every time we had a conversation, we're like, "Oh my goodness! We just learned all these stuff."

When we repeat some of the smart things that we learned from someone else to the next smart person, they'd be like, "I have these smart things to say, but those are some pretty smart things you just told us." We're like, "We didn't come up with it. We had learned it from someone else."

We said, "We should just put all these people together in a room, because all the smart people have figured out all these different smart things, but they're not talking to each other." That's literally how we started the summit.

[0:39:24.7] JM: Austin, you touched on marketing there. You were talking about the Doritos booth with the Cuban Sandwiches. I find the developer marketing stuff to be so interesting as well. You really see it in full force at some of these conferences. When you walk around these giant expo halls and you see five different cloud service providers, and five different monitoring companies, and they're all presenting a slightly different version of the world.

Let's talk about products. Let's talk about vendors. Richard, you're working on Datawire, and I know you're looking at a bunch of different product opportunities in the microservices environment, and there's so many of these different verticals. You got distributed tracing companies, and monitoring companies, and service proxying companies. Do you have a mental model for the taxonomy of the different types of vendors and where the good business opportunities are?

[0:40:29.9] RL: Yeah, I think those — The way I think about it is, at a bare minimum, if you want to do microservices, you need some way to develop those microservices. You need some way to deploy those microservices, and some way to run them. If you think about development, that just means a dev environment with all the tools, like Git — Brilliance, Git, and an IDE, and a debugger. Depending on how you like to develop right from a deployment standpoint. You need

a deployment pipeline that works well with microservices so you actually get your stuff running. Then, to run, you need some sort of container orchestration thing.

Then, as you get more sophisticated, you need service proxies, and monitoring, and that kind of thing. The question that we — When we talk to organizations, "What do you use for development? What do you use for deployment? What do you use to run this stuff?" As you get more sophisticated and you add more services, hire more engineers, what you need in each of these categories gets to be quite a bit more sophisticated.

Does that make sense?

[0:41:31.3] JM: It does. It's interesting, because a lot of these different area, they look like small markets at first and then as you zoom up close to them, it becomes a very big market, like, "Oh, we're doing distributed tracing for AWS." "Oh, that's what your entire company is about?" It's like, "Yes. That's all we do, and we make very good money doing it."

It's very interesting that there are these companies that can handle — That can take on everything. Even something like New Relic. When New Relic got started, people were probably like, "Okay. They just do monitoring?" Then, it's like, "Yeah, monitoring is a gigantic business."

Do you feel like the software company ecosystem, is it large enough to sustain the number of developer tools, companies that are coming up? Because a lot of them are — You get a venture funded business and they focus on one specific thing and you look at it and you're like, "Is that enough to provide venture returns?"

[0:42:34.5] RL: I think it's really hard. I think developers are so picky. Open-source is a terrible business model. Let's just get that all on the table.

The other thing that I think is interesting is I think that the answer to your question is, "I don't think it is, but I do think that there are opportunities. The reason why I don't think there is an opportunity for millions of developer companies is because developers are picky, open-source is a crappy business model. Most importantly, I think that 20 years ago you could make a pretty good business building infrastructure software.

By infrastructure software, I mean 20 years ago you had BEA WebLogic as an app server, and IBM Web Sphere as an app server and they made billions of dollars, and you could also buy ATG, or ColdFusion and probably 20 other things.

Today, no one pays money for an application server. Even if you try to create a company to make the next generation app server, there's going to be an open-source thing that's just going to get more community and move faster, or one of these guys, like Google, or Lyft, or Uber, they're going to open-source something and then they're going to destroy your business.

I think that, today, things have evolved to the point where it's really hard to build a company around infrastructure type software, because it has to be open-source, it has to be free, and you've got companies that are willing to just open-source stuff and keep on investing in it.

What I think that there is opportunity for is the value added layers on top of the infrastructure software, because at the end of the day, it's really still pretty hard to use.

[0:44:12.3] JM: What's an example of that? What's the value added layer?

[0:44:16.7] RL: A classic example is UI. Envoy has no UI. To configure Envoy, it required — In fact, Matt, actually, in the documentation, he wrote, "We wrote this other program that generates configuration files for Envoy, because Envoy configuration files are too complex." You're like, "Oh my goodness. You literally have to write another program to generate config files for this other thing that I'm about to deploy?" and your head kind of explodes.

[0:44:43.3] JM: Yeah. Then, it's like, "Then, we had to write a config validator for it."

[0:44:46.8] RL: Right. Exactly. You're like, "Oh my goodness! This is actually really complicated. Can I just get a UI for this thing?" I don't know if you can make money building UI for Enjoy or not, but I think that's a trend that you start to see is people starting to build UI on this type of stuff, or pre-integrating a lot of these different pieces together.

One of the things that we're doing at Datawire is we're helping companies take — Instead of saying, "Here's this massive ecosystem. Go build your own thing." We're saying, "We're going to take Kubernetes. We're going to deploy it on AWS. We're going to stick Envoy on top of it. We're going to pre-integrate a dev environment and a deployment pipeline for you, so you don't even really need to think about it. It's just going to do what you need it to do." Lots of companies like that.

That's kind of, I think, where you're no longer — We're not really writing a lot of infrastructure software. We're putting in pat, pull requests, and sending stuff upstream when we find a bug. We're really trying to give you value around how you tie all these stuff together to solve a bigger problem as supposed to, "Hey, here's the service proxy that you're just going to deploy."

[SPONSOR MESSAGE]

[0:45:58.3] JM: Don't let your database be a black box. Drill down into the metrics of your database with one second granularity. VividCortex provides database monitoring for MySQL, Postgres, Redis, MondoDB, and Amazon Aurora. Database uptime, efficiency, and performance can all be measured using VividCortex.

VividCortex uses patented algorithms to analyze and surface relevant insights so users can be proactive and fix performance problems before customers are impacted.

If you have a database that you would like to monitor more closely, check out vividcortex.com/sedaily. GitHub, DigitalOcean, and Yelp all use VividCortex to understand database performance. Learn more at vividcortex.com/sedaily and request a demo.

Thank you to VividCortex for being a repeat sponsor of Software Engineering Daily. It means so much to us.

[INTERVIEW CONTINUED]

[0:47:10.9] JM: The offering of we're going to layer a UI on top of Envoy sounds like a great lifestyle business, but it sounds like there might be a different capital structure that's needed for

that kind of thing. How does the venture model, or the fundraising world look today — How does that world look at these different business? Because if I'm a venture capitalist these days, probably there are 100 companies coming to me every week that are saying, "Hey, I've got this super —" I'm doing distributed tracing for architectures that use Cassandra." You're like, "Am I supposed to invest in that?" Maybe you just say, "No. Go talk to this micro angel investor."

Do you have a picture for how the fundraising market is going these days for microservice-based businesses?

[0:48:03.1] RL: I think that people are viewing it with — There's a higher bar for raising funding. By a higher bar, I think to get seed capital, you really need to show some proof of market momentum. It's not enough to just have a business plan or an idea. I don't think it's a huge amount of market momentum. They don't necessarily expect you to have millions of downloads, although that would be nice. It's just, "Hey, I've built this product and I have hundreds of users that I think can get to thousands of users if I could just put a little bit more money into a better blog. There're these 10 features that require a lot of work, like Windows support that I can't just do at night and on the weekend."

I think if you are able to show a little bit of momentum, then I think most investors that I know would be willing to cut you at least a check to get started just to see if there's something there.

[0:49:04.0] JM: Okay. Austin, you work in developer marketing, or engineering marketing. As you're trying to cut through this space and understand what developers are thinking or how to craft messages that will appeal to developers, what are some of the counterintuitive lessons that you've learned?

[0:49:24.6] AG: I love that question, because I have a really, really good answer that I've thought through about this. When I first started talking in Datawire, the Datawire CTO and architect, Rafi Schloming, pulled me into his office and he goes — He sits me down and looks at me very sternly and goes, how do you sell to developers?" That's not what Rafi's voice sounds like, but a little of dramatic emphasis.

I looked at him and I said, “Developers are really easy to sell to.” He looks at me like every developer will look at you when you say something that they don’t believe, because no developer believes that they’re easy to sell to. He immediately looked at me and like, “You’re full of shit.” That’s exactly what I wanted him to do.

Here’s the thing, is that developers are easy to sell to if you have a really, really solid product and it’s easy for them to get access to. If you’ve built something that’s valuable and solves problems, all you need to do is get their attention long enough and make it easy for them to read some docs and play around with it, implement it in a sandbox, or with a sign on, or get them starting on a small project, because if the tech is actually valuable, engineers being so truth oriented and all they care about at the end of the day is to do what you said it was going to do and did it solve my problem. If it does and they have a great experience about it, then developers are actually going to take care of a lot of the marketing for you, because you’ve built something that’s very, very useful for them.

By walking in the door and saying, “You guys are easy to sell to.” I’ve gotten their attention and made them want to prove me wrong all with one simple statement. If you say something about the technology that they also don’t believe, but that is true about it, then there’s sort of this immediate response of, “I don’t believe you, and I’m going to prove you wrong.” Because at the end of the day, if you’re an engineer building some of these architecture, you’re not going to take anybody’s word for whether it works or not, you want to actually see for yourself how it’s going to play with your particular app, and your particular infrastructure, and your languages, all that.

It doesn’t matter so much what you say as long as you get people connected to the technology. That’s where I think of it. That’s how I start with all these stuff. You have an honest conversation. You say, “This is what it does.” You pique people’s interest and then make it very easy for them to find out for themselves if it solves their problem or not.

As long as the product is powerful and useful, you develop credibility over by getting them play around with it and saying it does good things. The other thing is just being honest about stuff and being actually genuinely helpful.

Going back to the summit, Datawire organizes that, but the purpose of it was to get information from around the industry rather than pitch one thing or another. I think that's really key to being successful in a developer community. It's that counterintuitive. Just try and be — Don't necessarily come at it just from a revenue standpoint. Come at it from, "How can we solve really, really interesting problems?"

I was talking to someone at GitHub the other day about, "Hey, why can't we — If I started an open-source project, why can't I capture an email so I can follow up and talk to these developers?" Their response back to me was, "That's what the README is for." The fact that you can't email-gate something, we've used a feature for the developer rather than a bug for you, because if they find something that's useful, they're going to want to jump in your spot, or want to sign up for your list. Wherever you're at, they're going to come find you, because they're like, "Shit! This is useful. I want to use it."

From a developer marketing perspective, your job is just to grease the skids all the way from a product adoption standpoint as much as possible, and then make it easy for them to find you when they do have questions, because once they've got questions and they're getting value of it, the momentum starts picking up and it becomes unstoppable.

[0:53:43.5] JM: That's such a good point about GitHub. I think a broader point there is GitHub is not LinkedIn for developers. They could have tried to do that, but they basically put their foot down and said, "We would have conflicting incentives if we were GitHub for developers. Developers would see through those conflicting incentives and they would have a reason to go somewhere else." Yeah, I agree with you that developers are thrifty, skeptical shoppers.

We're nearing the end of our time here. I want to get a picture from both of you where you see the world of microservices going, and this is such a buzz-wordy question. I also think it's just basically — It's the equivalent question of asking where is backend software development going, but you're free to answer it however you like.

[0:54:35.4] AG: Let me take that first from Richard, because I think my answer is going to be shorter and less impactful. I think that, while my hope is that it becomes less buzz-wordy and it

gets integrated in it lexicon, and people understand that it has — Microservices has particular uses and also as particularly like non-use-cases.

I'm hoping some more nuance comes in there and then as more and more companies adopt this, I think we'll see those patterns begin to emerge. I'd like to see people have less of a knee-jerk reaction to it over the course of the coming months and years.

[0:55:14.6] RL: I definitely would echo what Austin says. I think, right now, we're getting into the point where people have a knee-jerk reaction on microservices as a buzz word. What I hope, and we're starting to see a little bit of this, is that microservices is now evolving into less of a term for an architecture, but more of a term around a methodology for building software. That methodology is about really rapid updates. It's about having small teams of developers actually working together inside a one big organization. Those are all principle that I think everyone can kind of get on board with. You want to get to continuous delivery. You don't want to have big centralized infrastructure.

Then, the question becomes, "What is the technology platform that you need to enable this?" There's a big cultural shift, and I think and hope that people will start to recognize that microservices isn't just about the technology, it's about the people, and the process, and they're going to start thinking about, "How do I apply this to my organization, because I do think that if it's done right, once you hit a certain degree of scale —" and that scale is actually pretty small. We're seeing companies with 10 developers even struggling with some of these challenges and seeing microservices as a solution for this.

I think once you sort of get to that point, I think you're going to see a massive increase in productivity, and I do think that this will actually change how people start to approach developing cloud-native applications.

[0:56:41.7] JM: Okay. Austin and Richard, thank you guys both for joining me today, and thanks for having me as a press member at the Microservices Practitioner Summit. I enjoyed the attendance.

[0:56:53.7] AG: We're really glad you get to come out, Jeff.

[0:56:55.2] RL: Thanks, Jeff.

[END OF INTERVIEW]

[0:56:58.0] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That's symphono.com/sedaily. Thanks again to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]