EPISODE 696

[INTRODUCTION]

[00:00:00] JM: Linux has created much more value for Google than it has for Linus Torvalds. Ruby on Rails has created more value for Airbnb than it has for David Heinemeier Hansson. Successful open source projects create more value than their creators capture, and that's one reason why collections of people on the internet are often inspired to work together on open source.

When an engineer creates an open source project and that open source projects finds a large audience, the engineer can often build a successful business. There are very good examples of this, like SpringSource, Cloudera and Elastic. These are massively successful enterprises that were founded by the creators of open source software. But in other cases, the value of the open source project gets largely captured by cloud providers that create a closed source version of the open source project and offer it as service.

Shaun Connolly has worked in senior strategic roles at software companies such as SpringSource, VMware and Hortonworks. Throughout his decades of experience, much of his time has been spent figuring out how to monetize open source projects intelligently. Shaun joins the show to talk about his past experiences building enterprises as well as modern issues such as how to compete with major cloud providers. We also discussed the Commons Clause License, a new software license that open source projects can use to try to protect their value from being entirely captured by a cloud provider.

Software Engineering Daily is looking for sponsors. If you're interested in reaching over 50,000 developers, you can go to softwareengineeringdaily.com/sponsor to find out more, and you can send us a message there. We'd love to hear from you. If you are an engineer working at a company that is marketing to developers or hiring developers, if you tell your marketing department or your recruiting department about us, you can help us out that way. You can just send them a link to softwareengineeringdaily.com/sponsor and we'll be happy to talk to them. In any case, your listenership is always enough to support us and we appreciate it. So let's get on with the show.

[SPONSOR MESSAGE]

[00:02:21] JM: I have learned a ton from QCon. QCon San Francisco takes place this year, November 5th through 9th, 2018, and I will be going for my fourth year in a row. I always love going and seeing the talks, and in between the talks I hang out and eat some mixed nuts, chat with other engineering leadership about the latest talks and stuff that they're seeing, the 50 different stream processing systems that we're seeing, the different databases we're seeing, and QCon is a place where senior software developers and team leaders and managers and senior leaders, they all gather together, and you have fantastic conversations. You have fantastic presentations, and it's extremely high quality. Its technical. A lot of it is also cultural and about management, and you can get $100 by using the code SED100.

QCon is a great learning experience. The presentations this year include 18 editorial tracks with more than 140 speakers from places like Uber, Google, Dropbox, Slack, Twitter. They are curated high-quality talks, and you can get $100 off if you use code SED100 at checkout for your ticket. Again, QCon San Francisco takes place November 5th through 9th, 2018, and the conference is the 4th through 7th, November 8th through 9th are the workshops. You can go to qconsf.com to find out more.

Thank you to QCon. If you are going to buy a ticket, please use code SED100, and we really appreciate the sponsorship of QCon.

[INTERVIEW]

[00:04:19] JM: Shaun Connolly, you are an executive product strategist. Welcome to Software Engineering Daily.

[00:04:24] SC: Thanks for having me.

[00:04:24] JM: Well, we have a lot to talk about. You've been working on software product strategy since the 80s and there's so much historical context that I want to unpack as well as talk about what is new and what is novel, and perhaps unprecedented. So I'm hoping we can blend some history and some predictions of the future. How has your approach to software product development evolved over the last 30 years?

[00:04:52] SC: I mean, just the technology space in general has evolved. I mean, when I first started writing software, it was when patriot missile radar systems, right? So very constrained memory environments. Then I was at Primavera Systems for about 12 years, project management software for very large scale companies. That evolved from basically PCs and DOS and went up through Windows etc., and the web era. But I guess since 2000, it's really been mostly internet-based technologies.

As e-business kind of hit the inflection points, a lot of great innovation started happening back in that sort of 2000 timeframe. We saw the rise of Linux, Red Hat Linux, things like that, and I think open source began to really take hold and drive a lot of the shared infrastructure services, internet services, general application platforms, things like that, which is I think where we saw a lot of the sort of infrastructure that powered the new generation of business. All the way through sort of mobile, internet of things, etc., that we see today. So it's been waves of innovation, waves of technology change and, frankly, these days, I think the change has been accelerating.

[00:06:16] JM: There's these different inflection points that you alluded to there. You've got the movement to just people having mobile phones, that's one thing. You've got cloud. You've got open source, the change in open source. You have borne witness to all of these things, and we'll go through some of the companies that you saw the exchanges firsthand, because each of these inflection points I think break rules and create new rules around software product strategy.

Tell me something counterintuitive about software product strategy that you believe today that maybe I wouldn't hear from somebody else.

[00:07:04] SC: I think how my product strategy role evolved typically when I joined JBoss in 2004, it was arguably just as open source software was starting to really pick up steam, so to speak. The conversations changed to how can you use the technology and become more familiar with it away from what's an open source license, right? Transition to really business is getting a lot out of it as supposed to looking at sort of the legal underpinnings.

I think the model, if you will, of openness, all the code is out there in the open, as a product strategist made my job a lot easier, because it becomes less about BS and more about, “Here's the technology. You could take a look at it yourself.” There's not a lot of overselling that happens in that space, and that's in many respect freeing, if you will, from product strategy perspective. Not only in your relationship with sort of the end users and customers of whatever you may sell as a business, but also partners and others who want to integrate with the technology, they can see it, they can touch it, they could feel it and they can get a feel for what the technology does well today and maybe where it's heading next, those types of things.

So I think that transparency is something that I’ve lived and breathed for practically 15 years, and I don't think there's a ton of people who’ve had that as transparent of a focus, if you will, from a product strategy perspective. That goes from application platforms like JBoss, to frameworks like spring, or even to full data platforms that Hortonworks had. It's not sort of specific technology focus that’s more your approach and your collaborative approach, if you will, with the customers and partners who may have an interest in a technology. That open approach really changes the nature of the relationship.

[00:09:15] JM: And that openness, that's in contrast to perhaps a more adversarial relationship between the software vendor and the customer, which I think that adversarial relationship was maybe the norm in early 90s, mid 90s, and that's why a lot of enterprises today have an allergy to lock in. They’re very suspicious of software companies. It's funny, because that suspicion now can sometimes undermine an enterprise that is trying to figure out what software vendor to go with, or if they say, “We really need multi-cloud, because we don't want to be locked in. It’s like, “Well, maybe that's a priority of yours, but maybe you should be trying to update your enterprise a lot faster so that you don't get taken over by competition,” and multi-cloud might slow you down, but it arises from this adversarial background with software vendors. Would you say that’s accurate?

[00:10:17] SC: I would say it depends on the vendor and there's different personalities, if you will, that are set from the top, but in general, yes, I think if you look at what drives companies, particularly the larger companies that are selling license software, in many respects they’re just trying to get the customer to buy the license, and whether or not they actually use the software, it’s secondary, right? So that buy before you're able to really truly use it at scale.

If you invert it in open source or even open source models that have commercial sort of wrappers around them, there's a clear way to sort of engage with and try out the technology and make sure it scratches the particular itch.

That changes the nature of the relationship to be more about how can I add value to the technology or the product that you're using, versus you need to buy my license, and that might be shelfware right? I think that – I've worked with a lot of engineers, I've been an engineer sort of in my past, and the thing I hate the most is to create features and functions the don't get used, right?

Shelfware is of no interest for my perspective. I want to create and I want to work with engineers that create great technologies and great solutions that customers actually care about and use. I think you get a good number of companies who have that type of relationship with your customer, whether you're commercial or open source. I think that works, right? I think it's a bit of what's the culture? What's the approach? What are you going to stand for from a vendor perspective? If you have it right, where it's more of a collaborative and open relationship with the customers, I think you have the chance to go further.

[00:12:10] JM: You said something interesting about open source accompanied with an open source product. If you think about yourself as the product strategist and you're trying to sell a company's product and that company is based in open source, you were suggesting that that makes your job easier. Some people might say that's counterintuitive, because we have examples of open source software that has been extremely successful where the company that's built around that open-source software is not able to monetize it effectively.

So there are some people who would say that an open source piece of software is much harder to monetize. What is it about that open-source openness that makes you more comfortable formulating a product strategy?

[00:13:01] SC: Right. I think we have tease apart a couple of elements. If you're a vendor-focused on bringing products and solutions to market that are largely based off of open-source software, there's a couple of ways you can go about it, right? How do you actually go about monetizing the software is different than the actual sort of technological approach itself?

So the collaborative open approach that a technology makes sort of driving roadmaps and engaging people out on GitHub and issue tracking systems that are publicly available, etc., to drive tech forward, I think alleviates a good bit of problems and opens the aperture to more inputs, which frankly makes a technology better more quickly.

With that, to your point, is you certainly can choose – There's a variety of models were it could be pure support. It could be commercial open core where you basically have commercial extensions to open source and there's a range of open-source businesses that are based on open source as a range of models, sort of the choose from. That, I would say, is aimed more and how do you envision yourself monetizing?

For instance, these days, if I'm advising anybody, you want to make sure you have – You've thought through if it's appropriate for your open-source technology between the platform space it is. What is your cloud strategy? How can you offer the technology as a service to end customers? Because in many respects, large IT shops are headed to the cloud and/or multi-cloud. So that has to be at least one facet of monetization.

At Hortonworks, we had cloud service offerings on Azure and Amazon as well as the downloadable software. So there were various ways you can interact with the technology and various ways the business could monetize that technology, right?

So I would say don't confuse monetization with sort of driving the roadmap forward and making sure the technology continues to sort of be vibrant and innovative and want to go forward. I think the open-source approach, particularly if it's a popular technology, helps facilitate quick roadmaps and a lot of interaction, which really helps the technology grow in the right way. I don't know if that makes sense, but that's kind of how I think about it.

[SPONSOR MESSAGE]

[00:15:44] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[00:17:53] JM: Let's go deeper on an adjacent topic. You and I were talking over email about the Commons Clause, and I know this is something that people want to hear about, and since we're already talking about open source and its relationship to building businesses, I think we should just jump into this topic. Can you just retell the story of what's going on with this Commons Clause? Give us some background in it is up to speed on the involvement of Redis

Labs and this company FOSSA. Describe what the Commons Clauses and why it sparked a lot of debate among developers online.

[00:18:38] SC: Sure. I think when you get businesses focused on monetizing particular to open source technologies is really sort of – If they're thinking about it right, they're always looking for a way to balance what's best for the developer, for the end user/customer and the vendor, right? So it's a balancing act, and there's a range of open-source models that are out there, and a range of open-source licenses that you can choose.

In the case of Redis Labs, Redis is a very popular caching technology and sort of memory oriented database technology, and Redis Labs, the company, has built extensions. They not only drive the innovation on Redis, but they built extensions around it that, from a commercial company perspective, they want to be able to monetize more directly. So they wanted to handle licensing of the extensions differently than the sort of the Redis core. There're a couple of ways you can do it. You could choose alternative open-source licenses, like AGPL, they could do more protections. In particular, they stated the concern with the larger cloud vendors, like Amazon, etc., sort of getting more of a free ride on the technology and being able to monetize it more directly than they would otherwise.

So enter the Commons Clause, which is basically sort of an addendum, if you will, that can be applied to a variety of open-source licenses, liberal licenses, like the Apache software license, which I'm pretty familiar with since Hortonworks is an all Apache projects. So the Apache license is very popular. It's a very permissive license, and the Commons Clause basically, simply put, adds a – If you're going to be taking this additional code, you cannot sell that code, and they apply that commons clause to the technologies that were more extension technologies, not to the sort of the Redis core stuff.

Optically, when you look at how it's applied, since it's just sort of an overriding piece of sort of legalese that says the code sort of will retain sort of Apache license, but we'll further restrict it with this cannot sell terminology. It's almost like you're trying to make the open-source license be commercially enough so you can address your business needs versus just detailing out a commercial license, right?

I think that's where it sort of ran a much, I think particularly in the open source world. There are very strong opinions that are out there, and I think it’s sort of the beehive was swatted, so to speak, and there was a lot of buzzing around the use of Commons Clause, and Redis Labs explanation as to why.

I’m not going to commend sort of directly on is it or isn't it sort of a good approach. I prefer more specific approaches where you have code that's open-source licensed, Apache, etc., and if I want something that has more commercial restrictions, then I would create a license specific for that code so people know what they are consuming, versus it almost looks like it's Apache, but it has this added clause to. It feels a little too hiding behind my index finger approach to licensing, which can get confusing. Being more explicit usually tends to be my recommendation, if you will.

An example of a company that was more specific and they may have had their and naysayers sort of out there as they’ve rolled out their licensing approach, but Elastic – With Elasticsearch, Kibana and others very, very popular actually gone public, etc. So very successful open source company. They created the Elastic license, in particular, to protect their expat technologies which are security and manageability monitoring and a variety of things around the Elastic technology.

But the Elastic license protects sort of the binaries but also enables them to put the code out there. So the code is in the open in GitHub, but it's not open-source licensed. It's using the Elastic license. In their case, they chose effectively a commercial license, but they also chose to put the code in public GItHubs so people can see the code. It just has commercial license terms as supposed to open-source license terms. That fell a little more specific, right? Two different approaches to a similar problem, if you will. I tend to prefer Elastic’s approach than the other, but I think that's kind of the net out, if you will.

[00:23:44] JM: If I understand correctly, the Redis example is Redis has an in-memory data, data solution basically. It's like a lot of people use it for object caching and other purposes, and the Redis project is open source, and companies like cloud providers will take Redis and productize it and make it a service. They’ll make minor adjustments to it, but they won't make contributions back to the open-source repository. So Redis Labs, which is a company that is built, I think, by the creators of Redis. Is that right?

[00:24:32] SC: Yeah, they’re involved in the project. I think there’s a bigger community. But, yes, they have people who work on Redis as well as the extensions. Yup.

[00:24:39] JM: Right. So their feeling is – Well, there's clearly some kind of market failure here if we’re making this open-source project, we’re contributing to this open-source project, and the lion's share gets captured by AWS in their Redis as a service offering. Therefore, we need to make adjustment to the license, the Redis license that says, “Going forward, if we update Redis, when we update Redis, the future versions of Redis cannot be used in for-profit,” or I don’t know exactly the wording, but basically it's a wording that said that suggests you cannot sell this. You cannot sell this is a cloud product. You cannot take this off. You cannot do a git clone and then repurpose it and sell it as a service.

[00:25:35] SC: Right, and those terms are really only applied to the Redis extensions, not to the redis core. So if people were creating in their other extensions that do some of the same things that the Redis extensions do, then they’ve created that technology and then you’re freed to do that.

Effectively, yes, that's what their goal is. One could argue, and I don’t have facts to back this up, just sort of industry hearsay behind it, but who’s monetized Elasticsearch the most from a business perspective? Has it been Elastic the company or has it been Amazon with their Elasticsearch service? Some say the Elasticsearch services has monetized it more. In the Hadoop space, you have Amazon Elastic Map Reduces done really well as well.

My point is, is that that's part and parcel of the open source bargain, if you will. You have to expect that others are going to monetize at least the core technology. In the case of Elastic, they chose to create specific license for some of their higher value ad that will control the ability of that higher value ad to be used directly by folks like Amazon and others. They can always form a partnership, and that's how that can be done right, but there has to be a conversation that happens.

Redis Labs is looking for a similar thing, and this is all really how do you get a balance of the developers of the tech, the vendor behind the tech and the end users and whether those end users are also other people who are benefiting from the tech, right? How do you get a fair exchange of value across that spectrum, if you will? It's not an easy problem to solve, right? The licensing is sort of a tactic towards that end, and what we find is sometimes if you take a more specific approach, people can appreciate that approach more directly, whereas you might get some uproar. I'm not saying one versus the other is any better. They're both trying to attack the same thing, which is give a fair sort of ability of those using the tech a fair chance of monetization, particularly the engineers who are developing it. That has been the case since open source started. So not a new problem.

[00:27:56] JM: Fair is a really tricky word here, because you see some of the arguments from the open source acolytes here, and some of them are pretty moralizing, and you have to wonder what does fair really mean here. So you have AWS, which is was first to market in this set of tools that dramatically reduced the costs to start a company, or to build whatever you want as a developer. This was tremendous windfall for developers.

Yes. Yes, open-source was also tremendous windfall for developers, which was bigger? Well, I don't know. You kind of can't have one without the other, but these moralizing people who just go really strong in one direction or the other, and Commons Clause is the death of open-source, or just people who – I think it's very hard to have a super strong opinion here, because on the one hand, it is kind of a shame that the creators of Redis or the company based around the creators of Redis would not going to capture a large share of the Redis market.

On the other hand, you could argue that we're talking about the cloud market here. If Redis is capturing a larger share of a subset of the cloud market, is that fair to AWS? Which was the first – Arguably the first real mover in the cloud market? So there's not really a moralizing argument to be made, I would say. I mean, how do we even evaluate whether this is good or bad? Do we just speak personally as like a developers or like advocates for the future? How do you formulate what is "right" here for you?

[00:29:49] SC: Well, the thing is, is Amazon is not doing anything wrong, right? The terms of the license that was chosen enables them to do what they're doing, right? It's not dissimilar to IBM over many years made a ton of money off of Apache Web Server and a variety of Apache Software Foundation technologies. They wrapped them up in commercial products that they sold the market, but those technologies were under the hood in their app servers, their web servers, etc., right? Was that bad? Was that morally wrong? No. They were basically consuming technology according to the terms of the license.

So, to me, this is where do you draw the line on where you focus commercial value ad if that's going to be your business model, right? Then what's the approach to licensing? I think it ruffled feathers on the "purity". To me, commons Clause is just another commercial license. We shouldn't get our feathers ruffled in that regard, because they're not saying that it's an OSI approved open source license.

But with that, optically, it can give the impression that it might be sort of open source-ish, and I think that's where you sort of get this gray area that all the skirmish happens in. Just don't lose sight of the fact that I think your broader point is increasingly enterprise customers want to consume things as a service, right? The move to cloud-based architectures, cloud native application development, DevOps, GitOps, whatever you want to call it, right? That whole wave is happening. Increasingly, more and more things are going to be deployed into infrastructure as a service, database as a service in and things like that.

I, as a company, if I'm building a company around open source technology, I'm going to want to make sure I have a strong offering in that as a service realm. To your point, there can be other players in that space, like Amazon already has a sort of a caching tier on that, that then puts pressure on you as a vendor to figure out who are you going to serve, how are you going to differentiate and how are you going to have a more feature-rich offering for the segment of the market that you want to go chase that maybe different than what Amazon is chasing? That's all in product strategy. That's all in how you want to build a roadmap that targets a segment of the market that makes both technical and business sense.

[00:32:31] JM: Let's continue focusing on Redis for a second. If you think about Redis – I mean, is this a way to build a durable company? Will this encourage the durability of Redis Labs' business model by adding the Commons Clause this open-source project?

[00:32:50] SC: Again, I think licensing is a tactic. What will be the success or failure of Redis Labs is what technology – What's their broader perspective on what they are building as a company above and beyond sort of the Redis core that's highly used by and loved by a lot of developers and a lot of use cases, right? So as a business, where are they going to add value around that? Are they just going to be a support company? Are they building commercial extensions to that? Then sort of tactically, if it's commercial extensions and/or managed service or cloud service type solutions, then how do you license it? How do you package it up? How do people pay for it, etc., comes after? But it really gets rooted in what are you focused on as a business? Redis is sort of a core technology they're going to build a business around, but that isn't the only thing, right?

I think the rest of these stuff comes as far as choice of licensing and that kind of stuff stop comes after, after the fact. That's why I use sort of Elastic as an example, is Elasticsearch Apache license very similar, widely used, very similar to Redis, and they've chosen a very specific Elastic commercial license to apply to some of their value add. But that license also enables them to make the code open. So the code isn't open source in the true sense, but people can look at GitHub and see the code behind the commercial software. To them, it was important, have the transparency in the source code itself even if the source code was commercially licensed and wasn't using an open source approved license. That is sort of more of an even tradeoff.

If you look at their business, they make money from cloud service type contracts, even though Amazon has an Elastic service out there. Elastic is doing similar, but they also – If download and install it, etc., etc., they have a multi sort of revenue streams out of that off a common base with sort of a clean license model for sort of the extensions that they deem commercial and they want to protect it a little bit more. So, as I said, that one feels right to me if I were sort of advising folks to go, "Yup! That's a pretty good model."

[00:35:24] JM: Right. May be one angle of looking at this is it's an unbundling to – Since we're in an industry jargon kind of conversation, to use some industry jargon, it's an unbundling of what you get with open source conventionally. Conventionally, with open source, we think of extreme examples, like the Linux kernel. But you could have other gradations. You can have – Like this code base is mostly open, but maybe certain components of it are non-monetizable, and the only reason that that seems allergic to us or to some people, is because of what their normalized to.

Now, maybe we could contrast this with some other open source sectors that are trying to find their legs right now. So a couple examples come to mind in the Kubernetes space. So you've got of a variety of Kubernetes as a service products that are not tied to major cloud providers, and they're kind of trying to build, in some cases, open-source platform, or system that's connected to Kubernetes. They're trying to make that work. There are also companies like service mesh providers, and it's an open question as to like how do you build a good service mesh business. Can you build a good service mesh business when a cloud provider like Google Cloud is just going to probably give it to you for free as a feature? I guess in your perspective on the world, do you have a vision? Do you have a differentiated vision on what a service mesh should be or what the features around a service mesh should be or what the delivery mechanism of a service mesh should be, and can you profit off of that vision despite the fact that your code is open source? So what's your perspective on the Kubernetes space and monetizing open source in the world of Kubernetes?

[00:37:24] SC: Yeah. So it's funny. I mean, the Kubernetes space has been playing on me. I also had Apache Mesos and a variety of other sort of platforms. When I was at Hortonworks, we were watching sort of the rise of container-based deployment and I would joke there for a while, I'd look at a magic eight ball to figure out who's going to win and I'd go, "[inaudible 00:37:45] again later." Then it became really clear about a year and a half ago that Kubernetes is going to win sort of the lightweight rail, if you will, for the new workloads, right? You've seen sort of the industry sort of place their bets around it, whether it's Amazon, Google, Azure Container Services, or whether it's Pivotal, or Red Hat OpenShift or what have you, right? Just been a phenomenal sort of coalescing around sort of the Kubernetes dial tone, right?

With that, you have other things, like service meshes and others, right? So Buoyant is one of the companies and I've have talked with William there, you bring ups service mesh, and there's a couple of ways you can look at service mesh as an example if you're trying to figure out how do you want to differentiate. One is you can view it from it's a feature of the platform, right? Platform owners who buy the platform are going to want a certain set of features around traffic routing, quality of service, security, things like that, or you can flip it and basically go, "What if the sort of the service mesh features enabled an experience for the service owner?" The people who are deploying the services have the beepers, and when things go wrong, they wanted be able to see the performance, they want to be able to interact with the health and vibrancy of the service, and it's less of a platform owner thing and really more of the service owner thing and enabling collaboration of the surface owner and the team who developed the service to be able to find and triage performance issues, whether it's services they consume, upstream, or whether it’s services they’re influencing downstream. Where does the problem lie?

One could argue, you could build a whole set of capabilities that appeal to a service owner that could be very differentiated than a set of capabilities that appeal to more of a platform owners trying to get consistency across every service. For a service owner, Linkerd 2.0 recently came out and I've seen some of the demos there, and it's easy just to mix in to your service and be up and running very quickly without a mother may I of what’s the standards sort of quote service mesh for the platform as a whole. The service owner is actually able to get a lot of value through dashboards and reporting and charts and things like that from their service perspective in a way that makes sense for them. The point there is it really depends on who you're appealing to.

Another example sort of outside the service mesh space to make it more concrete is application performance management tooling has been around forever, but New Relic built a very strong and publicly strong via an IPO business off of enabling developers who are creating apps to get introspection into how the app is performing and, oh by the way, a lot of those details were available on a software as a service platform where you can go view the performance, etc., etc., and they had built a very scalable SaaS based model off of that. I can envision service sort of sidecar and service mesh type capabilities being delivered through a SaaS model that sort of inject in in a more service owner focused. It really depends on which way you scratch the itch and for whom you scratch it for.

Sort of close out on it is one of my phrasings is; who are these people, right? Who are you building for sort of a Seinfeld thing? Who are these people? Who are these people? It's important to know who you’re building the tech for, right? Because service mesh is a general term, and in many cases, most folks view it from the platform up not necessarily from the service owner down, right? You could very much have two different solutions that have similar sets of capabilities but for different purposes.

[00:42:06] JM: Very true, for some of these vendors that are not major cloud providers, there might be some advantage to the fact that you are not a giant cloud provider, because there is a degree of distrust, the same thing that drives this multi-cloud concern I guess if you're a non-cloud provider. People will trust you more to be able to deliver multi-cloud solutions. So that does seem like a burgeoning advantage for some of these independent non-cloud provider companies, like service mesh providers, or a Mesosphere, or a Red Hat. Those are sponsors of the show, I guess I should mention, but I think that point stands.

Other opportunities in – I'm seeing this really big shift from – Obviously, the shift to the cloud, but there's also the shift towards Kubernetes, and these things are going hand-in-hand. If you're a giant enterprise, like an insurance company, or a bank, you are in the midst of adopting these technologies. You're in the midst of figuring out how to use the cloud. We're passed the world in which the cloud is a controversial move to make, because you want to use software tools that are only in the cloud. What are the other opportunities for new software companies to come up and how should they position themselves as we are in this moment of software architecture within large enterprises and the software buying patterns changing?

[00:43:46] SC: Yeah. So I think I always like disruptive and/or sort of looking at things from the opposite side of things, right? Before Hortonworks, I was at SpringSource and VMware. Before that I was at JBoss and Red Had, and the interesting thing on the JBoss to SpringSource realm was JBoss' focus back in those days was on J2EE application server platforms. So a very platform-centric approach, whereas Spring and Rod Johnson, whey were focused on how do you have an inversion of control? How do you invert how you think about the platform where I as a developer can have control and I can just sort of inject things in from the side and be able to move much more quickly and consume just the things out of the platform that I want to consume. Two sort of very different sort of approaches to sort of that technology, if we look at the cloud and that kind of stuff, whenever I look at a particular sector, if you're in a new space that the cloud vendors haven't gotten to, maybe that opens an opportunity to maybe partner with them and or come out with a service yourself around your time series database or what have you, right?

The other realm is, if it's just going to be stuff that's baked into the core, then you really have to ask yourself how can you be the best at something that either adds value on top of that versus basically trying to convince people that your thing, which is very, very similar to the other cloud thing is going to be good enough.

I do think there is value in a multi-cloud notions, like Hortonworks has a data plane service, which is all about sort of hybrid data management cross on-prem and multi-cloud and they're building specific features and capabilities that address that need. But in the case of data, data is everywhere and it's inherently a data is everywhere problem that needs to be solved. It isn't just on Amazon, or Azure, or Google, or on-prem, or what have you.

So you can't just say, "Well, I'm going to create something that runs in all three. There has to be true sort of value above and beyond that." That's why sort of the new relic thing is an example is that they start it off with sort of monitoring and that kind of stuff, which for the most part has been commoditized, but in their service you're able to bring a bunch of other data in there that gives you richer insights above and beyond sort of the application layer, but they started sort of in a particular area.

So you have to have a vision for as a software company, what do you have today? How are you building it out? What's the form factors? Cloud, downloadable or what have you? Then how do you envision? What are the couple of different ways that the technology might expand or grow and to a sort of an adjacent opportunities? Because the competitors [inaudible 00:46:51] cloud players or others are always going to be looking to launch, and you need to be like a shark. You need to be continually moving and swimming and making sure that you're picking sort of the right choice for sort of adjacent opportunities and higher value ad.

[SPONSOR MESSAGE]

[00:47:20] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[INTERVIEW CONTINUED]

[00:49:19] JM: I should talk some about your history. So as you've alluded to, you've worked at JBoss, at SpringSource before it was acquired by VMware, and that you are at Hortonworks for, I guess, more than six years, which is where you worked closely with Mitch, Mitch Ferguson, who is on to talk about go to market strategy a couple of weeks ago. I guess I want to talk through some of that. Oh! You're with him at SpringSource as well, I guess? You were VP of product management at SpringSource, and it got acquired by VMware, and this was in 2008, in the aftermath of the financial crisis. I mean, I can imagine that would affect any business to some degree. How did the crisis, the financial crisis affect the product strategy and the acquisition environment for SpringSource?

[00:50:16] SC: Yeah. It's kind of interesting, my experience at SpringSource ultimately led me to Hortonworks, and I'll sort of tell you the backdrop story to that, but basically I joined SpringSource in 2000. September of 2009 VMware acquired us. Frankly, SpringSource had served a million Java developers, right? So we had a very large, vibrant, happy community. So, in large part stuff that we added around it, I don't think we were really impacted much from certain financial crisis.

I think the marriage with VMware made a lot of sense, because Paul Maritz was CEO at the time, and he's a very developer savvy strategy-minded CEO, extremely impressive individual, and he saw this movement to cloud and really felt it was important to bring a the large community of developers and align them ultimately with VMware's cloud vision, which spun out to be Pivotal and Cloud Foundry and cloud native development and everything good that you see with Pivotal these days, right? That was a long game that Paul Maritz was playing at VMware.

I recall a meeting that we had with Fidelity, it was one of sort of our joint customers at the time before VMware acquired SpringSource where we're talking about his vision of platform as a service and sort of cloud development models, and one of the things that came up in that meeting was, "That's great. Higher productivity. Faster deployment," right? All the stuff that we know love as part of DevOps and cloud native development today was we were sort of panning this vision back. But they also said, "Come back when you've also solved the data problem, because you have latencies of getting data to the applications themselves."

Sort of fast forward to 2011 when I left VMware and I joined Hortonworks specifically to solve the data problem, so there are two hands, application developer productivity, fast turnaround, continuous deployment type development models and then also ubiquitous access to data and the ability to get data no matter where it is in a form factor that developers and analysts can get easy access to it. That was sort of the Hortonworks vision, and they were semi-related, whereas now let's remove the obstacle of the data, and that's where I spent the last 6-1/2 years at Hortonworks really trying to democratize data in many respects using an open source model.

[00:53:04] JM: Now, at Hortonworks, the Hadoop provider market in 2011, when I think back to that time, it seems like that was a fairly zero-sum market. So you have all these enterprises that are choosing between Hortonworks, Cloudera, or MapR. What lessons did you learn about strategy in a super competitive market like that?

[00:53:31] SC: Yup. I tend to be a pretty simple guy when I’m looking at what I'm trying to build, how I go about it and who am I competing with, right? So, for instance, when I was at JBoss, it was an application server, WebLogic had a highly rated one with technologists, developers, and IBM had it with WebSphere basically said, “Let's focus on taking web logic’s but, and we’ll win our fair share of the market. Don't try and compete with IBM, because they’ll do a top-down sale in our sales guys. We’re not selling software. We’re really trying to get it ubiquitous. We need to win the hearts and minds of the developers and the technologists.” I think that sort of won out.

In the case of Hortonworks, Hortonworks basically came into the market about three years after Cloudera. So they had been in the market for a while, and Mapr was also in the market ahead of Hortonworks. So that influenced sort of the model that we chose. Cloudera chose an open core model where CDH is open source, the Apache’s technology, and then security management, governance and all that stuff were commercially licensed, sort of add-ons around it.

When I joined sort of advising Hortonworks, I basically said, “You're not can it catch up and you’ll never break down the three-year head start. You basically have to make security management, governance, and not stuff open source and let the model kick in.” To the point where arguably Hortonworks is probably a little less than a year behind Cloudera from a revenue perspective now. So a few years were chopped off of that lead. Both companies were successful public companies. I think Hortonworks works will do over 350 million this year. So that's a lot in revenue and Cloudera is a little bit above that.

So it was just a choice. But at that time, when I was talking with Rob Bearden, the CEO, I was like, “Listen. We’re making a long – When we look at ourselves in the mirror five, six years from now, we don't want to – This isn't something you change. We’re choosing the open model for good or whatever. It’s going to – You can't waffle. You got to be true to who you are. That's what the brand is going to stand for,” and I think for the most part Hortonworks had done a good job of sort of standing behind that brand, being the open player, enabling partners like Microsoft and others to sort of get the value out of the platform and being a collaborative player there. But that set the tone of their personality and culture of the company, the roadmap of the product, the licensing strategy was kind of all related to that if you will, right?

I think as you pointed out, I was able to work with folks like Mitch Ferguson on establishing – I think, two things were important at the founding of Hortonworks. One was who were going to be our sort of cornerstone partners, and the two out of the gate that we chose were Teradata, because they were sort of the top of the pyramid “big data”. There was just a lot of very, very large enterprise IT shops that had Teradata and they would want sort of Hadoop data platform integrated with that, and then Microsoft, particularly around Azure HDInsight and the Hadoop as a service stuff that we did with them. Make it ubiquitous. Make it easy to use, two different choices on those. Both of those throw off decent and continue to throw off pretty decent amounts of revenue for the companies and they were commitments that were made pretty early on and took a while to build out.

The point there is my advice to anybody, is if you don't quite know what things are, make sure you think through and do your due diligence. One should take an approach. Be true to yourself. Be true to your customers, particularly be true to the community, because if you're just doing it for monetization purposes and only and not from an open source, true open source model perspective, you will be called out on it very directly down the line. I think we did pretty good jobs at that at JBoss and Spring and Hortonworks.

[00:57:58] JM: As we draw to a close here, I just want to give people some closing wisdom. You’ve been in product strategy for a pretty long time now. You’re advising different companies. What are the new kinds of problems that you are seeing in the companies that you’re advising?

[00:58:17] SC: Again. I’m engaged – My focus, really the sweet spot, is young technology companies that either have had seed funding, either self-funded seed funding or like series A. So they’re earlier in their journey, if you will. I think some of the things – Everybody wants to sort of pre-think it out and basically go, “I need to commit to a specific set of use cases and that kind of stuff and/or accelerate the product market fit, if you will.” I think there're waves of establishing a company. Number one is make sure you get a solid team in nucleus that you could build off of where you're passionate about a particular area. The next is how do you find sort of that product market fit, right? And that may take a while. It could be vertically oriented, horizontal, or what have, right? It really depends on what you're building and how you’re bringing it to market. If you’re bringing a cloud solution to market, you might aim that at small to midsize businesses and more of an enterprise downloadable thing by being the different folks, right? It really depends on sort of what you're trying to accomplish there.

Then once you have a repeat and once you have product market it and repeatable sales process, then how do you a scalable business and then really scale the revenue both with partners as well as directly? I think oftentimes people try and short-circuit through that. With clouds and SaaS, yup, you can potentially get faster to market, particularly with a hosted solution, but that has its own set of complexities where you’re signing up for 24 x 7 operational. So if you're down, your customer is down, right? As a service owner, you might want to have Linkerd 2.0 and it’s sort of service sidecar stuff helping you to introspect and find new problems, for instance.

I think that’s some of the things that I think are challenging. Open source is just a sort of distribution model, and I think that knocks down certain barriers to the technology being adopted quickly. I still think what it means to build sort of a scalable business, finding the right team, really finding the right product market fit. How do you package it for that product market fit? That blocking and tackling is pretty consistent in my mind, right?

I think building out the team, having engineer mentality that sort of embraces the business side, particularly when it’s time to scale. I think these are challenges that a lot of these young tech companies need help with navigating the waters, and that's what keeps me out of – Gets me out of bed in the morning.

I sort of closed with I think it's working hard, having fun while you work hard, right? Constantly learning and making sure you're sort of true to yourself in the process I think are sort of sort of defining things and how I approach matters. Do things in a very team-oriented way. When I engage folks and that, that’s some of the sort of the more subjective advice that I try and build out particularly as I’m engaging some of these young tech companies, is they need to work and act as a team. It's going to be hard work, but they need to have fun along the way, because at the end of day, you can really go sideways if you just get overly focused on one thing one way or the other, work, work, work without really keeping an eye on the sort of ultimate prize. So there's a balance there, and that's what I also try and advise folks to try and strike that balance.

[01:02:06] JM: Okay, Shaun. Well, it's been really good talking to you. We covered a lot of ground, and thanks for coming on Software Engineering Daily.

[01:02:13] SC: All right! I appreciate it, Jeff, and thanks for having me.

[END OF INTERVIEW]

[01:02:19] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]