EPISODE 756

[INTRODUCTION]

[00:00:00] JM: Infrastructure software can be a great business. An infrastructure software company sells core technology to a large enterprise such as a bank or an insurance company. This software has near zero marginal cost and it generates a large annuity for the infrastructure software company. Once a bank has purchased your infrastructure software, the bank is likely to renew every year and never remove the software. Selling infrastructure software is like selling concrete or steel, except the software is cheaper to produce.

It's easier to distribute and it generates annuity, rather than being a one-time sale. The fundamental economics of enterprise infrastructure software are extremely appealing and every year more businesses enter the space, but few businesses ever leave. If you're starting an infrastructure software company today, you can expect a complex battle for market share. There is no easy trick to get your software into the hands of your target customer.

Martin Casado studied computer science at Stanford before founding Nicira, a company that pioneered software-defined networking and virtualization technology. In 2012, Nicira sold to VMware for $1.26 billion. Martin now works as a general partner at Andreessen Horowitz. Martin writes about the modern strategies of building a successful infrastructure software company. He describes two methods of selling into an enterprise; bottoms up and top down. In a bottoms up model, engineers within an enterprise start using your product to solve a well-defined problem such as API management.

As more and more employees within the organization start to use your product, you can begin to engage this enterprise about becoming a paying customer for your product. Since the enterprise is already using your product, the sales conversation is much easier. In the top down model, you engage the CIO or the CEO or the CTO directly and you try to convince them that your product is worth paying for.

When the senior leadership of a bank buys into your product idea, you can count on that senior leadership to convince their developers to use your product within the enterprise, such as a

bank. It is a rare occurrence that your infrastructure software company will be able to fit cleanly into either of these models, bottoms up or top down. More often, there will be some bottoms up usage and some top-down buy-in for your product, but you will have to evangelize the product on all fronts. You will have to convince both the engineers and the senior leadership. Your infrastructure software product probably won't speak for itself. You will have to develop expertise in sales, marketing and consultancy. In many cases, you might end up in an unending chasm.

The unending chasm describes a mode in which an infrastructure software company must function as both a product company and a consultancy. Your consultancy is necessary to integrate your product into the enterprise and ensure that your software actually gets used, but it reduces the appealing economics of a pure software company with zero marginal cost. The unending chasm does not prevent you from being successful. Companies who have had very successful IPOs remain in the unending chasm today, but it's useful to know whether you are heading for an unending chasm or if you're already in one.

Martin Casado joins the show today for a discussion of product development, software engineering and go-to-market strategy. To find all 900 of our episodes including past episodes with a16Z partners, you can check at the Software Engineering Daily app in the iOS and android app stores. Whether or not you are a software engineer, we have lots of content about technology, and business, and culture, and investing. In our app you can also become a paid subscriber. You can get ad=free episodes if you don't like the ads, and you can have conversations with other members of the Software Engineering Daily community. I hope you enjoy this episode. Thanks for listening.

[SPONSOR MESSAGE]

[00:04:18] JM: Every developer uses open source dependencies in every software project. Whether you know it or not, your tools are automatically pulling in thousands of license obligations. You're also pulling in dependencies and potential vulnerabilities. This happens whenever you use a new component or simply run a build.

FOSSA is the tool for managing your open source dependencies at scale. FOSSA automatically scans your dependencies for license violations and vulnerabilities to prevent issues at build time

and automate compliance obligations at release. Over 8,000 open source projects and engineering teams at companies like Twitter, Docker and HashiCorp rely on FOSSA daily to manage their open source licenses and dependencies.

Get a free scan by going to go.fossa.com/sedaily. That's F-O-S-S-A, FOSSA. You can also check at the show we did with Kevin Wang, the founder of FOSSA, and hear how FOSSA can reduce your open source risks. If your company is good at developing software, then it should also be good at managing open source. Check out go.fossa.com/sedaily and learn how FOSSA can improve your codebase.

[INTERVIEW]

[00:05:48] JM: Martin Casado you are a general partner at Andreessen Horowitz. Welcome to Software Engineering Daily.

[00:05:52] MC: Hey, thanks for having me.

[00:05:53] JM: Your company, Nicira, was focused on making networks more programmable. How does network programmability today differ from when you started Nicira?

[00:06:03] MC: The story actually goes back a little bit before Nicira. So like a quick thumbnail sketch. I actually worked for the intelligence community, and this is like 2001 to 2003. If you remember back at that time, [inaudible 00:06:16] was getting pretty heavy. We are doing some pretty significant engagements. This is like post-9/11 type things, and we were pushing compute into environments that they kind of weren't created for, right? We're pushing compute into like nation-state type attacks. I was in the intelligence community.

At that time, what was really surprising was with compute, of course you could program it to do what you want. Actually, SELinux, for those that remember, came out of the NSA, right? So you could go into the OS. You could change the OS in ways that actually mattered depending on the threat environment. But networking and networking gear just wasn't built that way. I mean, networks were built in an era where we just wanted organic growth of the internet to cover the

globe and there was no way to really change the way you wanted it to work for things like let's say large data centers, or mobility, or security.

So when I pealed out of the intelligence community and I went to Stanford, that was kind of the focus of the research, which is like, "Can we change the internet architecture or the networking architecture in a way that it's more programmable?"

If you kind of zoom till now, which was your original question, I actually think the most interesting thing and what we kind of got wrong and then later got right was the network these days really is just plumbing. It just connects compute elements really, and there's no need to really program that. The functionality that you want to program actually has been consumed into the endpoint. So if you think about, for example, a data center, right? Instead of actually trying to do fancy stuff in the network, like say load-balancing, or security policies, or whatever, it's now consumed into either the hypervisor or something like a service mesh with Envoy.

So I think that the ultimate answer was not let's make routers and switches programmable. I think the ultimate answers is like, "Why don't we just pull this functionality into an infrastructure layer that's in software and running on x86 and use that instead?" and I think that has become the standard way of actually doing networking now, is actually in software at the edge.

[00:08:00] JM: When you're at Nicira there were several different business models that you iterated through. What did you learn from that experience of business model iteration?

[00:08:10] MC: Yeah. So at Nicira, we were at the cusp of going from basically on-prem deployment or people would pay for perpetual license to more service-like, cloud-like where people would pay for recurring. I made all the mistakes a naïve technical person makes in enterprise. Early on I assumed, "Oh! Wouldn't it be great if we just built a product and have somebody else carry it?"

Early on, for example, we had Citrix OEM, one of our products, which never really worked out. We tried to partner with everyone. We had early partners with like Google, NEC. A lot of the early clouds, we actually thought, "There's a partnership and we can get value from that." Then

later on we realized that, “Listen, this stuff is complicated. The best thing for us to do is to just sell it directly ourselves. If I couldn't sell it, nobody could sell it.”

So we ended up basically in a direct sale model of a product, and that was the business model that stuck, and it probably took us two years to figure that out. So the landscape was shifting and I think it's worth talking about that going forward. But as far as a business model, I think mostly they're just early mistakes, where honestly if I would've listened to Ben Horowitz, who’s on my board, I wouldn't have made them and we should have just started that way.

[00:09:19] JM: How do those lessons apply to 2018 enterprise software sales?

[00:09:24] MC: They partially apply. Here is what I think is the most difficult question to answer as a founder, and that is let's say things aren't working. Is it because you're not executing correctly or is it because you don't have product market fit? I mean, that single question I think is the hardest question for founders to answer.

Now if the business is working well, like great, things are fine. But so often you start a company and you get these small successes and it’s just not working how you’d want, and the temptation of course is to look internally and like, “What am I doing wrong, or what is the team doing wrong, or what is my sales team doing wrong?”

But it very well just could be that the market hasn’t arrived or you don't have product market fit. So I think the trick is if you’re founder is to execute in a way that you can actually answer that question, and that's the lesson that I learned going through this. So for example, I believe today, if the founder, the visionary, the person that understands the competitive environment, understands the market, understands the problem space, if that person can't sell it, I don't think anybody can.

So the problem of trying to take something and having like a third-party sell it or like doing [inaudible 00:10:27] relationship, the problem is not that those things almost never work, which they don't. They almost never work. The problem is more that if they’re not working, you don't know what the problem is. You don't know if it's because like the partner isn’t doing their work or the market isn't ready. So I would say from my lessons, I would say, today, for those that are in

the same situation as I was, which is it is a direct sale. It's not a bottoms up thing, and we should talk about that, and you're having a hard time, you try and sell it as the founder or as the founding team. Maybe you have a salesperson that understands like procurement, and only until you're able to get traction yourself should you think about expanding, for example, sales capacity by hiring more sales people. If people that work for you that you train, that you comp that are incented 100% of their time cannot sell it, fix that problem before doing something more broadly. Then eventually when all of those are working, you can think about engaging more broadly.

Now, that is for complex products which require a direct sale. A lot of the learnings that I had on that do not actually apply to a broader shift that's happening today, which is you're so much more kind of cloud-based bottoms up adaption, and that I am actually learning being on a number of boards and watching it and just being a student of that movement.

[00:11:36] JM: Eventually, you were acquired by VMware. Acquisitions are complicated. What advice would you give to 2012 Martin Casado going through the acquisition?

[00:11:46] MC: The best piece of advice I got during that, and I'm just glad it happened and I think that was the advice I needed to hear, it came from Diane Greene. So for those that don't know, Diane Greene was the founder of VMware and the CEO of VMware, and she sold it to EMC. Her and Mendel Rosenbaum sold it to the EMC, and then she exited the company. So she was an early investor in Nicira, and when I was talking – We were talking about, "Should we sell? Who should we sell to?" She gave me a call. It was a very short conversation.

She said, "Okay, Martina, if you decide to sell, and I don't know if it's the right thing to do, one piece of advice, and that one piece of advice is don't give up your sales team. The reason is, is VMware is very good at selling their people that run servers and they're very good at selling to compute, but those don't know how to sell to networking and security, which you're focused on. So no matter what happens, as far as the deal goes, make sure that you retain that." So not knowing and like being thrust in this situation, I actually held on to that really tightly. In retrospect, I think it was one of the best decisions we made, and certainly VMware made.

Going back to the previous conversation that we just had, it's very hard for somebody in a new area, in a category creating area to sell what you've done especially if they have no idea. So in the case of VMware, it would have been the core sales team. Could they sell it or not so? So we maintain our sales team for years, and because of that we're able to build a significant business, and now any sales rep within VMware is able to sell these types of things. But I think that was the single most important piece of advice, and it's one I would pass on to anybody in that situation.

[00:13:19] JM: When you were acquired around the time, AWS was very early. Since that time, AWS had become quite a dominant force. Cloud computing has been just growing and growing. It's been become really important. Did you predict that market shift that was going to happen or did it surprise you?

[00:13:36] MC: The speed in which it happened was what was so surprising to all of us. Everybody understood the model and everybody understood the disruption. What I could never have predicted is in – Listen, when I started selling networking software – It was basically network software. Instead of selling you're a box, I'd sell you software. You'd run in your hypervisor, not do working. That's basically what our product was.

When I first started selling that, pretty much everybody wanted to buy it using a perpetual license, as a piece of downloadable, installable software. Even this is when we were acquired by VMware in 2012. But three years in, almost everybody wanted this as a service or as a license. So for me, the biggest impact of things like cloud have been these follow-on effects, like how people assume they're going to consume software. How they assume they're going to pay for it. Not necessary that it's on-prem versus on-prem. That was – Even today, that's still not nearly as big as a shift as these other shifts.

So it surprised me how fast it happened, but that change is not that everything went up into AWS, it's running in AWS. It's the perception of how you consume software and pay for it was the shocking thing. I think, honestly, the entire industry today is dealing with that shift and still hasn't figured out, from investors, to founders, to customers.

[00:14:51] JM: In the regard to founders, the clouds are taking a ton of the opportunity. What are the opportunities that are left for infrastructure startups?

[00:15:00] MC: So I have two views on this. So in one view is the clouds are the new HPs, Ciscos, IBMs and Dell, and it's just infrastructure. It's compute. They'll do the most commoditized thing, and that's always been the case. But instead of like shipping you a box with the software on it, it's up in the cloud. In that case I would say, "Listen, there're tons of infrastructure to be built," I said on the boards of many companies building great infrastructure. So keep building great things horizontal plays, enabling the ecosystem where they're not, and there's a lot of value to be had.

There're a lot of successful examples of companies building core infrastructure just recently. Think Elasticsearch, think Confluent, think Databricks, think Knog. I mean, these companies have built real value. These are core infrastructure components in the cloud era, and they even host on the cloud and they treat the clouds basically as core infrastructure. So that's one view, and I believe mostly that view.

The other view is yes, they are like the old infrastructure players, yet they have this advantage that the old infrastructure players don't, which is they are such large point of aggregation. They can see everything that's going on, and therefore they can see into the future. If you have an incumbent competitor that can see into the future, they're much more dangerous than the old ones, which couldn't really see in the future. By that I mean they can look at everything, like AWS. Amazon can look at anything running on AWS, and if something looks like it's doing well, there's no reason they can't replicate it. Replicating that, especially in the world of open source, is simply just copying bits as supposed to having to rewrite it, or in the traditional model, then build out of salesforce and do all the go-to-market stuff around it, which is easily as hard as engineering often in new areas.

In this case, they just need to replicate it. A startup is already doing all the market maturation stuff and then they offer it for free, which is a much more dangerous situation. So I think there's truth to both stories. The first one that they're just infrastructure and they're focused on the 80% use case and not the new areas, but also I think they can respond to startups in ways that the

old incumbents could not. I think startups need to be very aware of that and have that part of their strategy to make sure that they stay protected or cannot move when these things happen.

[00:17:02] JM: If you are operating a large established bank today, what would your cloud native strategy be?

[00:17:10] MC: The interesting thing about the – So I've got a pretty specific answer to this. So the interesting thing about the banks is if you walk in any data center, their museums of computer science past. I mean, I remember one that we worked with. I mean, they had old SGI boxes, old Sun boxes. Here's what I think is the wrong thing to do. I think the wrong thing to do is to assume there is an operating model that covers all of those things, right? I think this is the biggest misstep I've seen when I was running, for example, large business unit. We dealt with these digital transformations all the time. They say, "Oh! The cloud is amazing. I'm going to do something in the cloud, or I'm going to do something cloud native, whether it's on-prem or off-prem. It doesn't matter."

But the mistake was, "In that model, I'm going to find some thin layer of software. Some orchestration system to apply to everything I have. So now all of my crofty, old steaming piles of 70s technology is going to be like easy to provision, etc." They all think – I means, actually, later they don't think of it as much. But early on, everybody wants to do that. They want to have like that one Band-Aid layer.

My recommendation is do adapt the most recent technology, and cloud is, to me, independent. Whether it's on-prem or off-prem. Cloud is whether you consume it as a service, the technologies that you use, the expectations around those technologies. Make that independent decision, but keep that decision in a silo, and things that are born into that go-ahead and deploy into that. But those that are not, that's okay. It's okay to have two or three operating models. It's much, much easier than having a lowest common denominator operating model that basically makes the cloud operate like your 1970s infrastructure .

[00:18:40] JM: And if you're laying out rules for the developers in your bank to procure software, you don't want them to have carte blanche to buy way too much software, but you

don't want them to feel restrained. What procurement policies do you put in place for your engineers?

[00:18:57] MC: I think I would worry more about like tool explosion and loss of development process, the natural procurements. I think the procurement rules are pretty good, which is more and more developers get discretionary spend. Sometimes it's individual developers. Sometimes it's their VPs of engineering or directors, and they can use that budget up to a certain point, and then after that, you have to go through procurement.

There're a number of reasons why a procurement office makes sense. Number one, like the professional negotiators, but more importantly they understand everything that's been bought from a certain vendor. So they can get maximum leverage on that. For example, if I'm going to get app dynamics from Cisco, I'm also spending $10 million on a VoIP system. Maybe I can get a discount on that, in a package on that or so forth.

So Procurement makes sense from that standpoint. Also, they know everything that's being used site-wide. So you can get again like these discounts, right? I mean, I just think from a sales efficiency procurement makes sense. So I think discretionary spending to some degree down to developers is perfect. I actually love this bottoms up organic consumption. Having procurement still makes a lot of sense.

But what I do worry about is like it's fine to use whatever editor you use. I literally like – If you've ever seen me do tweets or anything, I'm still using Vim. I literally – I still like use Vim plug-ins for email. I browse using like Vim key bindings. I'm just like a Vim person. That's fine. But for everybody to use their own workflow and their old tool chain I think is just disruptive to the engineering process.

[00:20:22] JM: Okay, and let's say the CIO of your bank comes into your office and says, "Okay, I'm going to an Expo conference next week. What are the kinds of companies I should go and evaluate and how should I make decisions around which of those companies to take seriously and to engage in a sales process?

[00:20:44] MC: This is where you're actually pointing out. This is actually – You're pointing out I think a piece where things are a little bit upside down, which is ultimately I find as the actual practitioners are the ones, the only ones, that can determine whether something is useful and how it integrates and the practical implications.

Often, the decision power rests high above that, and this can go wrong in so many ways. The people in the ground or something will be disruptive. Someone high up, say, a CIO, has their imagination captured and then a lot of problems could be avoided if they just would've listened. But like because the discussions were so independent, it could be problematic.

On the other hand, sometimes it is good, because especially us, we're so much closer to the code or what we're trying to do. Sometimes you have to be overridden by somebody that's like less risk-averse and has a broader view. This is all to be saying that, like I think it's very hard for a CIO to buy themselves in an expo, like get a sense of what's going to really work in an org, unless I mean they really understand their organization really seriously.

I, these days, would actually promote much more of a bottoms up approach, which is like let the engineers go to the expo, and let the directors go to the expo, and let the project managers and product leaders go to the expo. Let them come back and let them have kind of ideas of what's disruptive or non-disruptive, and then use that internally as a filtration function from that. I'm not sure that's the heart of the question that you're asking, but I do think that having sold all the way up and down the stack, down to the developer, all the way up to the CIO. I do think that the mismatch happens when those that have to implement aren't in the low.

[SPONSOR MESSAGE]

[00:22:37] JM: MongoDB is the most popular non-relational database and it is very easy to use. Whether you work at a startup or a Fortune 100 company, chances are that some team or someone within your company is using MongoDB for work and personal projects.

Now, with MongoDB Stitch, you can build secure and extend your MongoDB applications easily and reliably. MongoDB Stitch is a serverless platform from MongoDB. It allows you to build rich interactions with your database. Use Stitch triggers to react to database changes and

authentication events in real- time. Automatically sync data between documents held in MongoDB mobile or in the database backend. Use stitch functions to run functions in the cloud.

To try it out yourself today, experiment with $10 in free credit towards MongoDB's cloud products; MongoDB Atlas, the hosted MongoDB database service, and Stitch. You can get these $10 in free credits by going to mongodb.com/sedaily. Try out the MongoDB platform by going to mongodb.com/sedaily. Get $10 in free credit. It's the perfect amount to get going with that side project that you've been meaning to build. You can try out serverless with MongoDB. Serverless is an emergent pattern. It's something that you want to get acquainted with if you're a developer, and getting that $10 in free credit to have a serverless platform right next your Mongo database is really a great place to start. So go to mongodb.com/sedaily, claim that credit, and thanks to MongoDB for being a sponsor of Software Engineering Daily. I love MongoDB. I use it in most of my projects. It's just the database that I want to use.

Thanks to MongoDB.

[INTERVIEW CONTINUED]

[00:24:46] JM: So should the CIOs role be more of a curator of software that is bottoms up at a company that is naturally making its way through a company rather than a selector of top-down pieces of software?

[00:25:00] MC: I think it’s almost – I almost have a barbell view of this. On one side of the barbell, the CIO provides the environment where teams can choose the best technology for them and implement, and it's more of a meta-role. So it’s not a technology selection role. It’s not even really a technology role. It’s kind of a meta-role for infrastructure for the use and deployment of technology within the regulations and policies of the organization, which I always find CIOs to be much more about regulations and policies and information and much, much less about the actual kinds of nuts and bolts.

The other end of the barbell is organizations get stuck in local optima, right? I mean, one of the classic examples of this, and this is what I used to deal with all the time, is like there was a certain way of doing networking, for example, and all the practitioners listen. That's all they did

was they learned a certain CLI and a certain way of doing networking. They've got credentials based on that and it was very difficult for them to see how you can do with something else, because they're stuck in that local optima and they know all reason why something else wouldn't work.

For really dramatic drastic change, I think you need someone like a CIO to do that. So I think for the most part, you set the infrastructure for organic kind of bottoms up. Then on the other side, I think that if you need to do something so traumatic that requires basically top-down pressure, the CIO comes in. I think anything in between tends to have some sort of dissonance between the CIO role and the actual complete that know what they're doing.

[00:26:22] JM: There are some companies that are selling larger solutions with larger contracts into an enterprise, something like a Red Hat, or a Mesosphere. Then there are companies that are selling more points solutions, like a Kong, or a HashiCorp selling sets of tools. How did the strategies for go-to-market differ between these types of companies?

[00:26:48] MC: Yeah. That's a great question. More and more, if you have a product that you can sell bottoms up, it's a great way to get started. The reason is, is because it's actually very subversive to the incumbent stranglehold on an account, right? I mean, if you're a pipsqueak startup of 40 people, it doesn't matter how cool your product is. If you have your salesperson, they're having a conversation with the procurement office next to an incumbent who has hundreds of millions of dollars of leverage in the account, you're probably not going to win.

But if you can build a product that so much better than their product that developers and bottoms up adapted, there's nothing they can do to respond to it. The salesperson can't say anything against a co-product that a developer is using, and their product teams almost certainly can't make something like as flexible and as cool. I mean, that's what most technical founders are really good at, is making the coolest new popular product. I mean, they're almost like Shaolin monks that have been studying it for 10 years a second. That's like kind of the only thing they're really good at. You come out of school, you're really passionate about a product and a technology.

So it's very subversive from that standpoint. So I would recommend if you have a product that you can do bottoms up, you do that to start off with. However, at some point in time, enterprise sales tends to follow this traditional Pareto distribution where 20% of the customers have bought 80% of the dollars, or generated about 80% of the dollars. To get access to those dollars, you normally have to actually have direct sales and go through procurement apps so forth, and we've seen this with a number of [inaudible 00:28:13]. We've seen this with companies like GitHub or whatever, where they start bottom up phenomenon and then they take sales very seriously. I'd say Slack is another example of that.

So then you do want to shift to a more professional sales motion where you are going head-to-head with these other companies, but hopefully by then you're already a named brand. You already have account control and account penetration. You're already a known quantity. So you're doing it from a base that's also, again, one that's very subversive to the incumbents, because they just can't be as cool as from a branding standpoint, and it's still a fight to do it.

So there are very different motions, and sometimes you can only do them in isolation, but my recommendation just rather than doing that is to do both at the same time. One following the bottoms up and then the top down. Now, I mean your original question is how do these two differ? They actually differ and one is very much marketing-lead and inside sales, and the other one is actually direct sales, like hire an account rep going forward.

I just wanted to take that opportunity to make the point that –

[00:29:09] JM: The marketing one being the bigger solution.

[00:29:11] MC: So if you're doing bottoms up, like you're mentioning, like the early days of GitHub, or Slack, or Kong, or a lot of open source projects. I mean, the growth engine, the adaption engine is much more marketing-lead than sales, right? The product markets itself. You do meet ups. You do hackathons. You build open source community, or SaaS community. You use the products to create network effects and virality. You may use Google AdWords. You may have inside sales reps calling people to use it and so forth. But like because you're not collecting a lot of money per individual user using it, you can't spend a lot of money on sales. So

these are market-driven approaches. So that's very much the go-to-market for bottoms up. It's market-driven, marketing-driven.

Sales on the other hand is you have a very complicated product that if you did put it on a website, people would just look at it funny, which is a lot of infrastructure products. These are so complicated, like they don't sell themselves. They're not like self-explanatory. They're not cool. Then you have to do direct sales, where you pay somebody sell it, and those people at market rates for someone that can sell infrastructure software is expensive, like OTE, on target earnings. That's like their base plus their variable comp is 300 K+, often more. So you have to be able to sell it for a lot of money in order to recoup that cost.

So they're very different go-to-market models, like the mesosphere current direct sale go-to-market model and something that's more bottoms up, say, Airtable, which would be more bottoms up. But rather than have that be the dichotomy, like you have to do one of the other – I say like more and more. If you, let's can will start with the first one and then let's move to the second one.

[00:30:39] JM: In that answer, you gave examples of Slack, you gave example of Kong. Kong is a software company based on open source software. Slack is not based on open source software, but both of these are arguably bottoms up type of companies, because Slack starts with just some people at the company install it and they start using it and then all of a sudden it's very popular throughout the company. Might be the same thing with Kong. Some developers start using Kong as open source software and then eventually realize, "Okay, we have this everywhere in our organization and we need to buy the enterprise version of it."

So can you look at these two types of companies, the bottoms up close source company and the bottoms up open source company as basically in the same bucket or is there something about open and open source software as a bottoms up type of lead gen that makes it an open source company markedly different than a company like Slack?

[00:31:44] MC: I think this is one of the questions. I'm more and more of the opinion people over rotate on open source for the sake of open source, and what I mean by that is the real disruption to me that's going on is this bottoms up adaption and product selling themselves and

people consuming things as a service rather than buying things perpetually. I think I can come in a lot of forms. It can come in SaaS, like the case for Slack, where it's like a vertical services. They can come strictly as open source. I mean, Kong is actually not a good example, because they do have a service offering as well. But like let's say there's a company that only does an open source offering, and then they can come in combinations of them, like think Databricks, where there is Spark, which is a large open source component. But then there's also like a very large online piece that is a SaaS piece that's got a non-open-source components.

I think that's something that's actually quite difficult in offering an online service, which is how most people want to consume things these days. It's not just the code that does the basic thing they want to do. It's not just like Elasticsearch, right? It is all up the code to the operations of running a multi-tenant online service and doing all of that. I mean, that's easily just as complex to me, and that stuff is almost never open source. If it was, it's not clear how useful it was, because it's normally tuned to one operation.

So I would say whether it's SaaS alone, whether it's a combination of open source and SaaS, or is open source alone, all of those are viable approaches to doing this. But the big disruption and the big thing that they're enabling is allowing products to speak for themselves, allowing customers to get familiar with their product themselves without kind of like top-down pressure, like to integrate them, procurement that way and then growing organically that way, rather than having to fight basically a direct sales with an incumbent. That's to me the big disruption.

I think GitHub is a great example. Listen, git is open source. It's written by Linus Torvalds. It was a part of Linux. It had nothing to do with GitHub, but it's open source, and it drafted away. GitHub is an online service, which people can use for free. So there's a freemium component to it. It's fantastic. It piggybacks on an open source component, which is great, but that's not – If you actually look at all the code that goes in to GitHub, I mean how much of that is open source? May be a lot, but how much of that's actually developed by GitHub?

[00:33:52] JM: I mean, there's a sufficient amount that you want to pay for the close source offering.

[00:33:57] MC: Right. So what I would recommend to anybody looking at this, I'd say, "Okay, if you want to build something, some things that you build, people can only appreciate it if it's open source and they can see the source because they're integrating to that code. That's fine. Then you open source that."

Others, it doesn't really matter if it's open source or not, because that's not the use case. It's not a development use case, like Airtable. Think Slack. I mean, these are applications that you want to use, but no matter what, I would recommend offering something as a service. So if I'm the creator Elastic, I can open source Elastic, but I still want to do Elastic as a service, and once you offer something as a service, I think this discussion of open source kind of goes away, because it's so complex to develop the operations for one of these SaaS services. Even if you did open source it, like it's so tailored to that one organizational operation. It doesn't really matter. That's why you don't see people ask questions for SaaS services nearly as much as you do if like you're shipping closed source software.

So that to me is kind of my line of thinking, is like it should be SaaS. If people are integrating the code, it should be open source, but you should allow kind of cheap or free adoption so that people can understand the product and they can sell themselves.

[00:35:01] JM: So if I'm selling InVision, and it's it design tool, and it has a freemium offering, there's not much difference between that and having an open source version of Kong as lead gen for my enterprise software.

[00:35:14] MC: Yeah. I would say you're the reason that something like Kong has to be open source is really as developer, tooling. I mean it's part of the – It's often part of a larger application. It's front-ending the APIs. So actually having access to the source code is an advantage to whoever is using it. If you've got a product that's for a designer and they're not a developer, not integrating [inaudible 00:35:34]. I think just giving them access to products that they understand and they can use it and so forth is very similar.

I mean, I think one of the things that we've learned from open source is like just because it's open doesn't mean that you don't want to pay for it and have some stand behind it and work on it and so forth. I mean, this is just like inherent implicit in the complexity of code. So whether

you're giving away something for free or you've giving the open source for free. In both cases, there's so much value that the company can still add to it.

So then just the decision is, is like, "Okay, what should the product look like?" So it's almost a product design discussion. If the user wants code, you give them code. If the user doesn't care about code, you don't. But to me there's just nothing magic about open source. You just – One component of product thinking. It's not a rule. Does that make sense?

[00:36:18] JM: Yeah. Well, I think I agree and I think there's almost at the open source level, the products that do better tend to have a design ethos those around their repo and around their onboarding process, and then that comes down to more of a marketing decision and it's like how am I marketing this open source product to the actual developers that are using it? But again, I think it's – I'm increasingly convinced by the whole it's kind of just like a freemium kind of thing.

[00:36:46] MC: Just to see how I view this as an investor, which may highlight it a little bit, which is if someone comes in, a founders, and says, "I've got this project and it's a hundred percent open source, hundred percent open source and it's always going to be open source." That honesty doesn't sway me one way to the other. It doesn't scare me at all. I understand how complex it is to integrate open source and to maintain it and to do features and whatever. I just think it's impractical, like if they're solving a real need to think that like – I mean, unless it's super mature in a super mature space, say, something like nginx. But if this is something that's new and it's evolving or whatever, like it's impossible – I don't think it's practical to assume people are just going to take the open source and then they're going to be fine with it, right? They're going to want a company behind it.

So it doesn't scare me away if they say that. I may wonder like, "Okay. Listen, you're open sourcing some application for farmers. I don't think they care if it's open source or not. You can do it if you want." To me it's not necessarily a strong positive or a negative in of itself by its open source, but if I look at the context, then I start having more opinions on these things.

But I would like to reiterate the point before. I think the world wants to consume things as a service, whether its on-prem or off-prem. That's a different thing. Like you can still have a

service on-prem, right? That's just where geographically you are, but how you consume it and who handles the updates as who handles the operations, right? That's to me what cloud is, right?

Doing that requires a team full time all the code to run those operations. It's very deployment-specific, and those are the things that I think actually create the value these days, not whether something is open source or close source. So open source all you want. I mean, from an investor standpoint, that's fine. But I just think that's not the high order bid anymore. The high order bid is how you're delivering it.

[00:38:27] JM: Several companies with a core technology that is open source have IPO'd recently. You have Mongo, Elastic, Pivotal arguably. What lessons can the open core startups that are in their early days, what can they take away from these stories that have played out all the way to IPO?

[00:38:51] MC: So my big learning in all of these, especially if you look at those examples, is especially if you're selling infrastructure, you're selling a piece normally to a larger system, right? Whether it's a database, or it's a PaaS, or it's a search component, or whatever it is. In most infrastructure components, the interface is so sufficiently thick, and the system is so sufficiently complex that there is a ton of work to do to take the open source and to make it work within that broader system, and that's the value these companies end up providing, is the arbitration between their open source component and the larger system it comes into, and that can look in a number of ways. It can be close source components, which I'm increasingly believing like, "Fine, do it if you want," or it's because they provide the help and the services around that.

Now, services is traditionally been this kind of dirty word, where like, "Oh, if you've got a deployment, if you have people on-site helping with the integration and so forth, that's a bad thing, because margins are bad," but the reality is, is that people pay a lot for software. Margins is not just impacted by who you send out. It's how much you can charge for things. You can build these beautiful businesses where you are helping people integrate these open-source components into these larger systems and very few organizations want to do that themselves,

because they already probably have a negative unemployment rate. The technologies change too quickly and they don't have access to the code base to do it anyways.

So if I were starting an open source company today and doing some core technology, I would view how these companies did it. Because you build something that's very compelling, that's very useful, you engage with top-end customers, especially in these cases, to integrate it into their systems and you don't be shy about actually having bodies do that integration work and charging for it and just make sure that you charge enough to cover the cost.

[00:40:41] JM: How do you avoid that strategy degrading into an endless chasm?

[00:40:47] MC: The endless chasm – So I think the danger of that strategy is – So the optimal approach to that strategy, again, my opinion, is that on a per customer basis you can do integration work, but you're not doing core engineering work. By core engineering is you're actually fundamentally altering the product per customer in a way that it impacts every other customer and then you've got multiple products and multiple code bases and etc. That to me is the failure mode there. If you're doing just basic integration work where the core product says, "What are these?" but it's kind of at a higher – It's a higher level more scripting, may be writing new tooling around it and so forth. I think – Well, traditionally, it's that viewed as great. I think that's actually fine assuming that you can put enough money in it.

If you look at a lot of these companies, by the way, that are open-source companies that have been successful, you see that a lot of what they do is just providing support and provide these integration services. The endless chasm to me is just a term that I came up with when – The clouds crossing the chasm model which is a fantastic model. It's a fantastic model, but it's been around for a while, says that technology adaption follows a curve, and that curve says, early on, you've got innovators that will kind of look at anything, because it's cool and like they're really excited about it and they'll do it, and you can always [inaudible 00:41:59] to those guys and they'll kind of try anything, just because it's technology.

Then you go to basically the early – From like the innovators to the early adapters, and the early adapters also like they'll try things because they're cool. They're maybe like less experimental, less crazy, but maybe more plugged into the business than the experimenters. Td then you've

got what's called the chasm, and the chasm is the difference between the market treating what you have as a foreign object to a known object. The way that you build your company depends on how the market treats your technology. So the hope for a company is that you're following it like a classic technology adoption bell curve, where at some point the market matures to the point that it's an unknown object. Now it's much easier to sell that. It's much easier scale as your sales force, like the discussion changes to more one of like kind of benefits of yours versus somebody else's, verses like you need this versus you don't need this.

By the times it's matured, people know that they need it and so forth. So that's kind of been the hope and the guidance to build the company. The problem is I found now that the market is so dynamic and technology changes so quickly and buyers so often can't even keep up with the trends. That is not clear that these technology adaption cycles play out like they used to play out.

I mean, you could – If there's a great new technology, you're just entering the "chasm" and then the landscape changes, and now all of a sudden you got to let go and start all over somewhere else. So many companies I see, it feels like they're in this endless chasm where the market is shifting so rapidly. The competitive environment is shifting so rapidly that it's just not – Well, the model may hold. It's just not useful to them. It's not useful to say, "Here's what it should look like if that's never how it actually looks," because things are changing so often.

So I think if you believe that things are different now than they were before and that model isn't as useful in these environments, if you believe that, I think it's worth thinking about, "Okay, what can you do as a company in order to navigate that?" I think there are many things you can do. I think one of them is – One of the implications of that is the customers are confused. So if you can be basically the buffer between the customer and the changing technologies, you're adding value that way.

So now you're the one that's syncing the dynamism of the technology adoption market. So you basically can go and say, "Hey, listen. I got you. You don't have to worry so much about the technology adaption curve. I'll worry about that," but then you were in a more of a consultative phase and you're the one that has to be able to adapt to the shifting technology changes.

I also think like things like how incumbents. We touched about this early on, like how incumbents respond is much different than it was before. I mean, it was funny. I was having this online debate with Geoffrey Moore, who is the creator of the chasm model, Crossing the Chasm. Tremendous work. We actually had him come and speak with us at VMware. I'm a huge fan of his work, but under the model, and he says this verbatim. He's like, "Listen, incumbents can't compete with startups in early areas because – In pre-chasm areas, in category [inaudible 00:45:10]. Incumbents won't compete with you, because A, they can't execute at small scales, and B, they don't want to. Like nobody cares about a $50 million market if you're the size of Amazon.

But unfortunately today that isn't true anymore. Like you and I were talking about, if Amazon sees you do something even if it's a small market, it's not very – And it's open source, it's not very hard for them to put something up that can potentially compete with you. So I think it's worth it to start acknowledging that can happen, and then to protect against it, which I absolutely think you can. I just think it requires a different thinking, a traditional thinking around technology adaption curves.

[00:45:41] JM: So if I understand correctly, what you're saying is that there is more and more software to be adopted, and if you're running a large bank, it's hard to figure out how to even consume that software even though you know you want to. So it makes sense for these companies who are in a "endless chasm" to look more and more like consultancies, because they want to bring on more and more service integrators to help the bank consume their software.

[00:46:12] MC: Perfect, and this gives me a good opportunity to point out too like some like second-order trends we can look at that kind of validate this opinion. So in some areas of infrastructure, the amount of dollars spent on services is twice or three times the amount of dollars spent on actual product. So you just want to know where the bulk of the dollars are going. A lot of them were actually going in services.

I think like, for example, the telco service dollar spent alone is 80 billion dollars. In my line of work, there aren't a lot of 80 billion-dollar markets out there. If you look at the companies that are servicing these, I mean they're fine at what they're doing, but they're not the correct teams,

right? I mean, think Capgemini, DiData, Infosys, they're all great companies, but they're generalists, and they solve lots of problems for lots of companies, but they're not particularly experts in cloud native, or Kafka, or databases, or the most recent technology that comes up. They tend to be experts in much older technology, like the long tail of traditional technologies.

So where do we stand right now? Let me just, from a macro level, pencil out the situation. You've got technology that's changing faster than we've ever seen. You've got a negative unemployment rate in a lot of these areas. It's really difficult to get talent. Then if you did you, you'd probably be training the people full-time anyways, and then all of the dollars as a result, many of the dollars, [inaudible 00:47:39] dollar is going to like helping people do the integration, all of these heavy work, but the companies that do that, their DNA has come from doing kind of much more traditional like legacy type deployment.

So it seems to me that there's a very nice spot for a company if you have a new technology to both be your own integrator and provide that. I think the trick is to mean internal discipline so that you keep your product your product and you know what dollars go for integrating your product. So over time as that matures, you can move that out into the partner ecosystem to do that services work and then you just focus on your product stuff overtime. But that only works once you hit maturity and if you hit maturity.

[SPONSOR MESSAGE]

[00:48:29] JM: If I were to interview for a software engineering job right now, I would fail that interview. The skills that you need to do well in a software engineering interview are not the same skills that you build in your job. This is a weird paradox within the world of software engineering, but it's a reality, and we have to cope with it.

Software engineering interviews challenge you and show many areas, algorithms and data structures, databases, systems architecture, the ability to talk to people. There's a time limit. There's whiteboarding. It's completely different than working as an engineer, which is why so many people study intensely for their interviews. If you are starting to look for a new engineering job, consider the App Academy Engineering Interview Prep Course.

App Academy is deeply familiar with the software engineering interview process and their curriculum is curated from over 30,000 engineering interviews. The engineering interview prep course is an online class that will get you up to speed on everything you need to know to get a better engineering job than your current one.

Go to softwareengineeringdaily.com/interviewprep and get $100 off the online course. Software interviews can be stressful and hard to prepare for. App Academy's engineering interview prep course will help you build your skills and build the confidence that you need to do the sorting algorithms, the binary tree questions, all the material that you've forgotten since your last interview. Go to softwareengineeringdailya.com/interviewprep to get $100 off the online course and put yourself in a position to get a job that you are more satisfied with, and a higher salary.

I am so glad that I don't have to do software engineering interviews right now, because I'm a podcaster. But if I were going to go back into the field and I had to do all those crazy whiteboard questions, the App Academy Engineering Interview Prep Course would be quite useful. So you can go to softwareengineeringdaily.com/interviewprep to find out more.

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

[INTERVIEW CONTINUED]

[00:50:53] JM: Some questions about other markets. There are a number of second layer cloud providers. So I've talked to some companies recently, like Spotinst, or ZEIT, or Heroku. How did the margin profiles of second layer clouds compare to those of base-level clouds, like AWS?

[00:51:16] MC: I do think that you are asking the right meta-questions, and the right meta-question is, is this a winner-take-all market? Is there enough off scale level mode? Is there enough of a scale level mode, because you have visibility and because you've got the economies of scale that you end up becoming the next generation telcos.

So what pretty much any economists will tell you is that if you have an economic system where the incremental cost of adding another user converges on zero, this is absolutely the case with telcos, right? Then you'll end up in a monopoly situation. That's just basic economics. It's

nothing to do with technology, right? If the incremental cost of adding that additional user converges on zero, you tend to have monopoly situations. Then the question is, is that the case for these large clouds?

Now, it seems to be that clouds can actually differentiate – By the way, that services is a commodity service, right? That's something that anybody can offer that service. It seems like the clouds can differentiate and add value. I mean, the workloads that run on Azure actually tend to be quite different than those that run on Amazon.

[00:52:12] JM: Or DigitalOcean.

[00:52:14] MC: So we're investors in DigitalOcean also. Tends to be much more developer-focused, different types of workloads. So I think actually you may end up in a situation where you can maintain margins through differentiation and through fragmentation of not the customer base, but the use case base.

It's interesting. We used to sell on multicultural environments all the time, but multi-cloud didn't mean the same workload [inaudible 00:52:34] between clout, which I think is one of the biggest misunderstandings in the planet. Just because someone is multi-cloud doesn't mean that they can actually arbitrate clouds. It just means that they use multiple clouds for the use case that's best suited for it. If you maintain that, I mean this a trillion dollar market. It's all of IT that's going after it. Then you can think like actually there will be a fragmentation, there'll be differentiation if that's the case. You will be able to maintain margin.

But it's early to say that for fact. You could also go back to the telco guys and say, "Actually, everybody is going to offer the same thing. The incremental cost [inaudible 00:53:01] user is going to converge on zero. Therefore, the largest one will win," and right now that looks like that would be Amazon."

[00:53:06] JM: Why hasn't there been a company that does this arbitrage between different cloud providers?

[00:53:13] MC: Will, so many have tried. I've been very close to a number that have tried, and here's my view. My view is because it's not really solving a customer pain point, and if that's your primary value, I don't think you can build a successful company. Now, if it's a secondary value, it's great, but I don't think you can sell on that alone.

So how do you save money on Amazon? The best way to save money on Amazon is you get big enough that you can get an account rep and then you negotiate it, so like you just have a lot. Another one is you can do things like rightsizing, like companies like Spotinst. I mean, what they do is they just got very good predictive models on pricing. You've got dynamic pricing and they'll do – They'll kind of arbitrage that for a single cloud, right? But it's not clear to me and I just think like precedence is now shown us that there just isn't enough value, and I don't even think there's enough customer willingness to try and run two different clouds, two different operating models that are optimized for two different things for price differentiation, especially if these prices are like toughly competitive anyways and converging on the same pricing model.

So I remember in 2007 when there are companies popping up that were going to be like the hypervisor for clouds and like the medical outlier and like you can run the same workload and all of them. It never played out. Different workloads go to different clouds. For the most part, the majority of workloads, I'd expected to continue to be the pace. Now maybe there would be some like tale of workloads that don't that are going to actually be hopping between them. But I think pricing falls an entirely different trajectory and I think you save yourself a lot more money by negotiating with Amazon directly, and/or using something like Spotinst or any of these companies that actually do good predictive models on pricing.

[00:54:52] JM: Question about the API economy. My sense is that companies get started and they're doing something, and whatever they're doing they'll look for APIs to solve problems within that company. So you start an Uber and then you say, "Okay, I need to solve the background check problem, you find an checker. I need to solve SMS, I find Twilio." There needs to be less analysis of the API economy when companies are in the ideation phase, but it seems like you could just go on to Rapid API or some other API aggregation service and look through APIs and find ideas to start businesses, API mash ups. Why hasn't that been a big trend? Is there something deficient in the API economy today? Have we not hit a world where you can just stitch together APIs and build a business?

[00:55:39] MC: I think we're there now. So I think the reason we weren't there 10 years ago, is I just believe the market wasn't large enough. So listen, the people listening to this may groan when I say this, because it's such a hackneyed thing, but I just have – I actually just believe it to be true. So here's the hackneyed expression, which is you know what? As markets get larger, the granularity for which you can get a business gets higher. So like, again, the most hackneyed like story ever, but it's worth saying, is Ford. Early on, when the market for cars is a few hundred cars, they had to do every aspect of production. They had to like – I mean, they literally have like the Rouge River complex for Ford. They would take in water and rubber and coal and like iron ore and out would come cars. They have to do every aspect of that. If you compare that to, now, the multi-trillion dollar car market today, you've got third, fourth tier suppliers that will supply nuts and bolts and screws and aftermarket cameras, and each one of those independent components are independent companies potentially.

Compute, we've seen a number of these also expansions. Originally, the only way you can sell a computer system, like let's say you're like thinking machines, or your IBM way back when, [inaudible 00:56:43] is like you literally had to like – You're responsible for the chips, the board, the sheet-metal around ii, all the software on all the applications on it. You come a little further to like the PC era. Then, like, okay, software was decoupled from hardware, but you still like – The OS and the OS provider did most of the things. You come up a little forward, then independent applications became independent companies. Like ERP got decomposed into like CRM and databases.

So now I think we're at the stage where an independent application is being dilapidated, like the marks have become so large that you can take an independent application and you can plot independent functions, and those functions can be entire companies. If that function is sufficiently complex and has commonality between applications, there's no reason somebody can't do that.

I've been in the Bay Area for 20 years. I've driving up 101 and back for those 20 years, and like what's on the billboards is often what's on the mind of people, right? And it is, right? I remember in like the early 2000's, it was bet.com and Yahoo and it was like .com stuff. Then I remember a little bit later on, it was like a lot of SMB stuff, kind of Barracuda and this and that. Then a little

later on it was all the social networking stuff. Remember that? It was like Friendster, this and that. A little later on it's like all the mobile stuff.

Now, if you go, like you've got a AI, IoT, but you often have these like developer companies on the billboards, right? It's pub/nub, it's Braintree, it's Twilio. I remember the Twilio one is like, "Ask your developer for API companies." So I just think the market has expanded to the point. It's large enough now that you can build a company that's just services and API. If you look at like Rapid API. I sit on the board. I'm an investor of Rapid API. I mean, they have like nearly 10,000 APIs that do everything from sending SMS messages, to hard-core AI, to like making your text look like Yoda said it, right? I mean, like there're so much out there now. So I do think we're in an era that you can build an application and a lot of the hard parts are available via APIs.

I mean, I head a statistic. So I just want to be very clear. I don't recall exactly the number, but I believe it was something like, on average, if you download app from the mobile app store, it will do 16 externally APIs. Whether that's SendGrid. Whether that's Google for identity. Whether that's Twilio, right? So that's a lot, and I think that number is just going to increase. Now, listen. We should double check that number, but like it's on the order of.

[00:58:58] JM: Sure. Your philosophy around company building emphasizes survivalism. You like markets where survivalist foundry can wait for the market to evolve into their vision. Why is survivalism a useful trait for a founder?

[00:59:12] MC: It's a great good news. It's one of the two – I actually shamelessly stole this from Chris Dixon, but like when he said it, it rings so true. I think the thing is either like die quickly or survive, but I actually focus on survival. I think it's probably a better way of saying it, but having gone through what I went through, like I really empathize with those on the survival site.

So Chris Dixon said this very well. He's like, "Listen, if you look at a lot of really hard spaces, the ones that made it out of the other end are the ones that just simply survived and waited for the market to takeoff."

I mean, like for example, in cryptocurrencies alone. There were long periods of things not really working out and people got frustrated and they left, or they ended up spending too much money and having to shut down. But the ones that are able to survive, when you had the upswing, do the upswing with it.

You can argue this is going on the drone market right now, right? Some markets are very hard. They take a long time to figure out, and so it's much more important that you're ready there for the inflection of the market and to somehow think that you can create the market, because creating a market is incredibly hard.

So in my case – Listen, is that I started the company in 2007 where literally like a viable business model was like you're a smart Stanford PhD. That was my pitch to investors. Then a year later, like the world ended. Man! It was 2008. This is the worst economic times since the Great Depression, and you couldn't raise any money. Mean! We stayed 12 people for years, and I looked around and like – I mean, we had a lot of potential competitors at that time, and every one of them died.

I would say a large portion of our success is we just survived that like really meager, crazy downtime and then we were able to draft on the upswing where people become more optimistic and they try new technologies and budgets do come up. I just think – So I just do think that for hard markets, it's the right approach. But then you don't want to become a victim to your own strategy. When it's time to take off, you have to take off too, and that's hard if you spent years of belt-tightening and like an ascetic lifestyle of IKEA tables and $4 lunches and paying yourself minimum wage, like I did for years. To turn that on and like to actually like invest in the company and burn a lot of cash, it's hard.

Anyway, I just didn't want to be very categorical and I wanted to put context on that. But I do think it's something for like if founders are frustrated and things are hard, I think like it's good to really belt-tighten until the market takes off. It's very hard to create a market yourself.

[01:01:27] JM: Martin Casado, thanks for coming on the show.

[01:01:29] MC: I really appreciate it. Thanks so much.

[END OF INTERVIEW]

[01:01:34] JM: Kubernetes can be difficult. Container networking, storage, disaster recovery, these are issues that you would rather not have to figure out alone. Mesosphere's Kubernetes-as-a-service provides single click Kubernetes deployment with simple management, security features and high availability to make your Kubernetes deployments easy. You can find out more about Mesosphere's Kubernetes-as-a-service by going to softwareengineeringdaily.com/mesosphere.

Mesosphere's Kubernetes-as-a-service heals itself when it detects a problem with the state of the cluster. So you don't have to worry about your cluster going down, and they make it easy to install monitoring and logging and other tooling alongside your Kubernetes cluster. With one click install, there's additional tooling like Prometheus, Linkerd, Jenkins and any of the services in the service catalog. Mesosphere is built to make multi-cloud, hybrid-cloud and edge computing easier.

To find out how Mesosphere's Kubernetes-as-a-service can help you easily deploy Kubernetes, you can check out softwareengineeringdaily.com/mesosphere, and it would support Software Engineering Daily as well.

One reason I am a big fan of Mesosphere is that one of the founders, Ben Hindman, is one of the first people I interviewed about software engineering back when I was a host on Software Engineering Radio, and he was so good and so generous with his explanations of various distributed systems concepts, and this was back four or five years ago when some of the applied distributed systems material was a little more scant in the marketplace. It was harder to find information about distributed systems in production, and he was one of the people that was evangelizing it and talking about it and obviously building it in Apache Mesos. So I'm really happy to have Mesosphere as a sponsor, and if you want to check out Mesosphere and support Software Engineering Daily, go to softwareengineeringdaily.com/mesosphere.

[END]