EPISODE 742

[INTRODUCTION]

[00:00:00] JM: Most applications today are either deployed to on-premise environments or deployed to a single cloud provider. Developers who are deploying on-prem struggle to set up complicated open source tools, like Kafka and Hadoop. Developers who are deploying to a cloud provider tend to stay within that specific cloud provider, because moving between different clouds and integrating services across clouds adds complexity.

Ben Hindman started the Apache Mesos project when he was working in the Berkeley AMPLab. Mesos is a scheduler for resources in a distributed systems allowing compute and storage to be scheduled on to jobs that can use those resources. In his time at the AMPLab, Ben collaborated with Matei Zaharia, creator of Apache Spark. Ben founded Mesosphere based off of his work on Apache Mesos, and since 2013, Ben has been building a company to bring Mesos to market.

In the meantime, several market forces have influenced the world of enterprises. Enterprise businesses built on virtual machines and on-prem hardware are trying to migrate to containers, Kubernetes and Spark. Cloud providers like Google and Microsoft have risen to prominence in addition to Amazon's continued growth, and enterprises are increasingly willing to adapt multiple clouds.

I spoke with Ben Hindman at KubeCon North America. Today, the company that Ben cofounded works to provide tools for managing these changes in infrastructure. In our conversation, Ben and I talked about the necessary mindset shifts that he had to make when taking a research project and turning it into a highly successful product. We also talked about the newer trends in infrastructure. Why enterprises will want multi-cloud deployments and how serverless APIs and backends will make the lives of developers much easier.

We are currently looking for sponsors for Q1. If you're interested on advertising on Software Engineering Daily, you can go to softwareengineeringdaily.com/sponsor. We're also conducting a listener survey. You can go to softwareengineeringdaily.com/survey and you can find that survey. We would love to know what we're doing wrong, what we're doing right, and you can

also take a moment to enter your email address in that survey and get some Software Engineering Daily swag or get a chance at getting Software Engineering Daily swag. You can also sign up for our newsletter by going to softwareengineeringdaily.com/newsletter.

With that, let's get on with the episode.

[SPONSOR MESSAGE]

[00:03:01] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW]

[00:04:24] JM: Ben Hindman. Welcome back to Software Engineering Daily.

[00:04:26] BH: Thank you. It's great to be here.

[00:04:27] JM: You created Mesos at the Berkeley AMPLab.

[00:04:31] BH: Yup.

[00:04:31] JM: Ion Stoica was recently on the show. I guess he led the lab, and you worked with him when you were creating Mesos. Also in the AMPLab was Matei Zaharia, who created Spark. What was it that made the AMPLab so special?

[00:04:47] BH: That's a great question. What's interesting is the AMPLab was the home for the projects, but we were just pre-AMPLab when we first started working on this stuff. We're actually part of the RAD Lab. So at Berkeley, I think it's done an amazing job as an institution of basically creating these multidisciplinary all within computer science usually, but multidisciplinary across various parts of computer science labs where they bring all these teams together and you've got folks that do machine learning, you got folks that do systems, you got folks that do hardware architecture. You got folks that do programming languages all come together to figure out how they can possibly solve problems.

But before the AMPLab was the RAD Lab. After the AMPLab is now the RISELab, which I'm sure Ion talked about on the show. I'll have to go watch that show – Listen to the show. I think that's the first real ingredient of what makes these special, what makes it special that so much great tech can come out is you just blend people of various interests and backgrounds together, and from diversity I think you can get some really, really powerful stuff.

In our case, what's interesting about where Mesos came from, where Spark came from, they actually came – I was actually working originally in the ParLab, which was another one of these labs. Again, Berkeley is really, really good at these labs, and the ParLab was the parallel computing laboratory. So I was doing parallel computing and the folks I was working with, Matei, other folks, they were doing distributed systems, and parallel computing and distributed systems are very, very similar. In fact, it's interesting when some of the first work I was doing actually had 128 core chips that I got to play with. At the time, the promise was that we'd have 128 cores in our laptops and our phones. We're not there yet, but hopefully one day we'll get there. The distributed systems folks, they had clusters and in some cases I had more CPUs in my single 128 core chip than they had in their entire clusters, which I suppose was always a fun boasting or bragging thing to talk about.

So the first ingredient was I think bringing this diversity, bringing a bunch of people together. Then I think the second ingredient is the labs tended to be funded by the industry. So there is an

exposure to problems. There are exposure to real problems that needed to be solved. I think oftentimes people talk about academics being in their ivory towers and just working on whatever they want to work on theoretically or otherwise. I think all the work that comes out of Berkeley is great, but I think one of the things Berkeley is really well known for often times is helping to create industries, because there's a good connection to the industry and the problems that the industry is having, and then you combine that with a bunch of diverse folks and smart folks and think about problems, and we solve things in ways that can end up being consumed and used.

In Mesos' case, we were solving the problem of how do you manage all these distributed systems. It's complicated. It's not easy. You got a ton of resources. What's the best way to manage them? The same parallel computing problem I was solving before, but I just transferred it really from doing it in a parallel computing setting to doing it in a distributed setting, and thus Mesos was born, and Spark came out of, "Okay. We've got MapReduce, but we'd really love to do things more iteratively. Let's not reload the data, reread from disks every single time. Even if we get SSDs, it's still going to be faster if we'd stick it in memory." That was a real problem, and thus Spark was born.

It's actually pretty fun. Spark actually was created at my parent's house in Colorado, which I don't know if Ion talked about that. But it's a fun old story. Yeah, I mean the reality was, again – Now the AMPLab, we had teams working in machine learning and they were coming to the systems folks with the problem of, "Hey, it takes so long for me to run this iterative job. Can't we just solve this with a better distributed system?" We said yeah, and the student at the time that was asking about that and talking to us about that was a guy named Lester Mackey, and Lester has gone on now. He's a statistics professor at Stanford, and he was doing great research, but he just wanted to speed it up, wanted things to go faster.

When we first conceived a new distributed system to make this go faster, we actually called it Lester's framework. That's how we tokened it. It was Lester's framework, and there's this really fun photo that whenever I get to connect up with Matei and Andy and some of the other folks from the AMPLab, RAD Lab, we sometimes joke about that photo, because that was beginning of Spark, was building this framework for Lester and making it go faster.

[00:09:38] JM: Interesting. So when you're in this lab, at this point there's been a cycle of luminaries that have gone from academia to industry and back. You've got people like Michael Stonebreaker and Ion. Ion started two very successful companies. You've got Dave Petterson in the mix. What's interesting about these folks is they not only have the computer science academic distance from the problems. They've seen so many cycles that they know that everything new is old. It's been around – The same problems kind of just returned in different forms. So they have a certain – At least this is something I've noticed from computer science professors that have been around for a while. They have this ability to take a distance, a distant approach.

When I was undergraduate, I would talk to some of these computer science professors and I'd be like, "Oh, you don't know what you're talking about," and it's sort of like I guess as you get a little bit older you're like, "Oh, I guess I should respect the elders. They know what they're talking about."

What have you learned from those kinds of folks who have spent a lot of time in industry and in academia and who have that long sense of remove from problems?

[00:10:57] BH: Yeah. I think it's critical. We often talk about, "This is an evolution or this is revolution." That's a phrase that sometimes gets thrown out there, and when you're doing research, is it just – Is this kind of the next step or is it just an evolution or is this pretty dramatic? Is this a revolution? Are you changing the way in which people should think about doing stuff?

I think oftentimes in academia, I think a lot of people are going after the revolution. I think academia is sort of affords you the ability to do that, because when you raise money, raise grants, you get money from the government. There's clearly a definition of failure, which is you were able to do very little with that money. But you might fail to achieve exactly from an academic perspective what you're trying to build, but you learn so much along the way and you publish a ton of great papers and you gave that information back to the scientific community that they can then build on in future work.

So I think in academia you're well-positioned for really that revolution perspective, and that's a distance I think. It's like taking a step back and sort of seeing that distance and trying to go do something there. But I think to me what really sets apart Berkeley professors often times from a lot of other professors is how they can connect the revolution to the evolution, because they have been an industry. Again, Berkeley is unique in a sense that it puts these labs together, which are funded in many cases by folks in the industry, the Googles, and the Microsofts, and the VMWares and all these companies, and I think that helps create this really perfect storm of revolutionary thinking, but almost connected to evolutionary how it can be applied and used in industry, and Berkeley is very well-known for, as I said earlier, helping to create industries and helping to create tech that is adopted and used in serious ways throughout the industry. All the way back to the UNIX days and plenty of other examples.

I think the thing – I guess the thing I've learned to answer the question is that if you want to make a big impact, you should start with that revolutionary thinking, but you need to also apply some cycles into how do I actually make this connect back and apply to how people can actually use it today. I think that's really the crux. Again, I think Berkeley is great at that and they continue to do it, and now with the RISELab I think there'll be other technologies that will come out of it that people will be able to use.

Going back to your earlier question as well, what makes Berkeley so unique. I think the other thing is there's an attraction to Berkeley from a graduate student perspective. I know I had this when I went to Berkeley, but there's an attraction to being builders, wanting to build stuff. Again, I think it's because of Berkeley's long history of building stuff that's a bit more revolutionary, but can be applied in industry. I think that's really attractive to a bunch of people. Not everybody, but I think you tend to get folks that choose Berkeley over, say, Stanford, or MIT or some of these other schools, because they want to go be the builders of those things. They want to go build that thing.

Even that as a mindset is I think a critical one to actually get your ideas out into the world and being consumed, because if your mindset is just the extent of what I'm trying to do here is come up with a bunch of great ideas that I will disseminate via research paper and go talk about at a research conference, it's going to be very, very hard for those things to actually get picked up in any kind of industrial setting, or at least it might take a very, very long time.

[SPONSOR MESSAGE]

[00:15:01] JM: Software engineers can build their applications faster with the use higher-level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin up cost-efficient, short-lived infrastructure. Container platforms let you scale your long-lived infrastructure for low cost, but how do you get started with all of these amazing tools? What are the design patterns for building these new types of application backends?

IBM Developer is a hub for open source code, design patterns, articles and tutorials about how to build modern applications. IBM developer is a community of developers learning how to build entire applications with AI, containers, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job. Get help with side projects, or think about how new technology can be used to build a business. A softwareengineeringdaily.com/ibm, you will find the resources and community who can help you level up and build whatever you imagine.

Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications, and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless, and containers, and artificial intelligence APIs.

So thanks again to IBM Developer.

[INTERVIEW CONTINUED]

[00:17:00] JM: Over the holidays, I talked to my older brother about this a bit, because he does research, and he started going to come conferences where they talk a little bit about building business around your research. He does it in the biology area. So I think it’s a little bit different

than computer science. It's not quite as easy to – If you've got a project in academic biology that you want to like take to market, I think it's a little bit more complex, at least from some point of view.

So Berkeley primes you with that builder's mentality. It sort of says, "Whether you want to do this in academia or industry, here's an environment where you can build something with an application." So when you move from the research side of things to the application business side of things, did your mindset have to shift? What kinds of shifts did you have to make to go to the business world?

[00:17:51] BH: I mean, yeah. 100%. Your mind has to shift. I think it depends on the role you end up playing as well. If you into industry, things are still very, very different. First, there is actually a pretty big jump still to this day from what gets built in an academic setting versus how that's going to get run in an enterprise setting. That's a huge jump. From academia to enterprise, it's massive.

Little things like did you build this thing from day one with the perspective of multi-tenancy including? That's a massive one. In enterprise, there's going to be a ton of users using whatever you're doing. So did you build it that way or not? Did you build it in a way where you thought about security and who can do what and how things are going to be communicated? It's massive. It's security. I mean, I can't tell you to this day how few systems in academia have security from day one. They just don't. But it makes sense. As a grad student, even if you're a builder thinking about how you want to architect this thing, are you going to spend the extra time thinking about what the security architecture is going to look for? You're not. You're going to work on getting it to have the best performance to solve the problems you're trying to solve as a systems building, as a systems researcher. Then you'll circle back and put security in.

To be perfectly honest, I think that's even more of a failure of our programming languages and our frameworks, our software frameworks than I think it is of anything else. I think if we had software – If we had programming languages which force this stuff upon you just like we can force type systems and program in other particular ways, I actually think that we could solve these problems, but that's future research that people could potentially do.

I think there is definitely a shift in thinking and the work that people need to do to transition. It's interesting. Again, if you into industry and you're working at tech company where your job is to just build tech, I think it's a little easier to stay in the mentality of the builder mentality and to think a little bit revolutionary, but you're probably going to be thinking more evolutionary quite frankly, because even if you're in a tech environment, you have your own internal users now, and your own internal users, they ask for far simpler things than what you might want to build, or they might ask for something and it's easy for you to say, "Oh, I could build that this way," this great revolutionary grandiose vision for how to solve those people's problems," and you say, "Yeah, it's going to take me 6 to 8 months." Then they'll say, "Well, couldn't you just do this one little thing and then I can do that," and you're like, "Yeah, I could, but then I'm not really innovating. I'm just evolving this thing." I think that's a big challenge for a lot of people. I see that every single day with engineers all the time.

But the best products quite frankly are the ones that really do evolve, that they kind of show up day one from a revolutionary perspective, but then they just evolve with the customers in a really, really nice way. The example I like to give often for that is the iPhone. The first version of the iPhone, which did very well. It didn't have copy-paste. It just didn't have copy-paste. Now anyone who's using any modern cellphone, mobile phone, we take it for granted that we would have that. First version, no copy and paste. That's okay, because that showed up in a subsequent, the next version of iOS. I mean, that wasn't like revolutionary. That was just like a pretty minor little thing, but that's where I think the best products are just – They're thinking about those little things and they're putting those little things and they're making that all be a great experience.

That's definitely a different mindset. Thinking about that next thing that you're going to just – That next small thing for the customer I think is a totally different mindset than the academic mindset of what's a revolutionary thing I can do. The revolutionary iPhone thing for copy and paste would be – I don't know, something crazy. Like you can squint at the screen and it notices exactly which text you might be looking at and it says, "Oh, that's what you're trying to copy." They would have just done something crazy, but that would be more the academic perspective. It's like, "Oh, copy-paste. That's easy. That's just a bunch of code we need to write. We just haven't written it yet," but that's what you got to do. You just got to do those things. That I think is a big transition and is a big change.

On the business side, honestly, it's even bigger. For me, it's been an interesting journey, because as a cofounder, I'm not just doing tech stuff. In fact, it kind of goes in cycles. At the very beginning maybe I was still doing a lot of tech stuff, then I spent time doing a lot of business stuff, business strategy, and there's a lot of things that are applicable from the academic perspective, but the reality is, is the business world is a different world. You think differently. You have different goals, really, that define what success looks like, an academia success looks like, introducing a bunch of great ideas. In business, ideas don't pay anyone salary. Often times you'll find yourself having the company invest in what you might not even consider to be amazing revolutionary ideas, but there's a market out there and you know people want to buy those things or will buy those things and it's just a totally different mindset to actually take a step back and say, "No. No. No. I don't need to think about the most revolutionary thing. I need to think about the thing that the customers are going to want to be buying that they need and what they're going to be doing."

That's why I think a lot of businesses have – Under CTOs, they have things like research, because I think if you just did that, if you were only constantly thinking about what the market needed today and the product you're trying to build for the market today, then you would get – When the next great thing came out of academia or anywhere else, you wouldn't be ready to jump on that and take advantage of it and make it be part of your company, or you wouldn't yourself be the ones that are doing the innovation, being the ones that are doing that. I think businesses know that and that's why they create these research arms and there's a bunch of really, really good tech that can come out of those research arms.

It ends up being a balance. Again, I mean, if I go talk to some of my customers and I say, "Hey, I know you want copy and paste and it's going to make your life so much better, but like that's just kind of par for the course. So we're not going to build that, but we're thinking about this crazy other future, blink copy." I know what my customers would say. They'd say, "Whoa! Whoa! Whoa! Whoa! Whoa. No. No. No. Just give me the simple thing. Just give me copy and paste."

[00:24:43] JM: All right. I think a perfect example where you've gone with the company, because you started out with something that was revolutionary. Mesos providing cluster scheduler, data center operating system stuff for people, and you found traction there. The

market has gone to a place where the focus of enterprises today, as far as I can tell, is that they want an ability to manage multiple open source frameworks, which does fit into your original vision, but there's also the focus on how do we manage a bajillion Kubernetes clusters. It's a focus on Kubernetes and it's also a focus on how do we adapt cloud services and multiple cloud services from different providers and how are we going to manage all the billing and how are we going to manage all the different accounts, and it's just kind of a disaster right now.

People have no idea what to do. If you go and talk to different vendors who are selling to them, the vendors can give them some sense of vision, but the visions are different from vendor to vendor. You have enterprises walking around and they're kind of – Some of them are a little bit paralyzed. Some of them are just like gradually easing into their buying decisions. Tell me a perspective there. How do you position yourself in this super crowded market where you have all these different kind of distributed systems modernization vendors trying to compete for the enterprise contracts? All the vendors are offering different visions. How do you position yourself?

[00:26:12] BH: Yeah. Yeah. I think you're actually capturing it pretty well for us, which is tons of Kubernetes clusters, but not just Kubernetes, a bunch of other services, distributed systems, data services people are trying to run as well thrown into a mix of, "Do I do this myself, or do I do it myself on-prem, or I do it myself in the cloud, or do I just let the cloud do it all for me?"

Actually, it's a pretty fun time from a business perspective when it comes to that, because you'll even hear this from AWS and the leadership at AWS. It's still the early days for cloud. It still really is. There's a lot of organizations that are in the cloud, but it's still early days and there's a huge opportunity to help people with that transition.

The thing we focus on, and I'll actually give you our vision for the company that we'd say to all new employees and it's kind of elevator pitch in many ways. What we want to do is provide a public cloud experience or public cloud services from an open ecosystem. So from open source software, open source distributed systems but that you can run on any infrastructure. You can run anywhere. You can run it on-prem. You can run it in the cloud. Because the reality that we've seen is going from 10 years ago, if you were a company 10 years ago and you decided you wanted to go to the cloud, the cloud 1.0 of our world was – It wasn't services. It was virtual

machines. We went to the cloud to get virtual machines. That's what we did. We went to AWS and we got EC2 instances. Then on top of those EC2 instances, we did whatever we wanted.

The reason why we did that was because it took 30 seconds to get an EC2 instance, but it took six months to get a physical machine in my own on-prem environments. Even if I had something like VMWare, it probably wasn't 30 seconds, but it still probably took some amount of time, because I still got to go through some tape to be signed off that I can actually get a virtual machine. Early push, the cloud 1.0 push for people, was all about getting VMs fast. Then on top of those VMs I'll do whatever I want.

You had companies, though I like to think about it – Everyone kind of looks at Silicon Valley and they say, "What Silicon Valley is doing now is what enterprise will be doing in five years." There's some truth to that. It's not black and white, but there's definitely some truth to that. 10 years ago when you saw all these Silicon Valley new companies just going to the cloud to get EC2 instances, now what we like to think of as the cloud 2.0 of the world is people don't go to get EC2 instances. I mean, some people still get some instances, but more and more what people do is they just go take advantage of the services. They don't go get an EC2 instance and then stand up a message queue. They just go to AWS and get Kinesis.

They don't go get an EC2 instance and stand up MySQL. They just go RDS. This is kind of this cloud 2.0 works, which is people want these services, but it's a conundrum, because if people go and they just use these services, they'd lock themselves in. They get the agility and the speed so they can start building their apps and going, going, going, but they ultimately get themselves locked in. I think there's enough people out there that are worried about all the lock in they've had in the past with all the various software that they've used. So they're trying to figure out what they can do to actually get them so that they're not going to get locked in.

They want to be able to leverage the cloud. Ironically, they want to leverage the cloud from the perspective of just getting those EC2 instances and then they decide what's on top, but they want as a service simplicity, the push button simplicity that they would have just gotten from the cloud 2.0 world. That's where we really come in. That's where we come in and where we position ourselves, is our platform gives you handful of these services which have all been built

in an automated way, automated when it comes to install, automated when it comes to upgrade, automated when it comes to scale up, scale down, automated when it comes to security.

We've put all the automations in to make this system on top, whether it's Kafka, Cassandra, Elastic, Spark, Jenkins, a bunch of others. We've put the automation in place to give people as close to that public cloud experience, that gets back to our vision, the public cloud experience of just clicking some buttons, but from an open ecosystem, all that tech that I just talked about. They're all open-source software. Anybody can just use those APIs in a completely open source way, but on any infrastructure. They can stand us up on any of the clouds. Again, getting back to just using the VMs, or they can stand us up on-prem.

That gives those organizations a lot of the power and control that they really want, because infrastructure operations organizations, I mean the struggle that they have every day is they don't want to keep their teams from moving fast, but they don't have the tools in their tool belts internally within their own shops to be able to stand up a Kafka in days, or minutes, or stand up a key value store, or a database, or an analytics cluster, or notebooks for doing machine learning, and you're going to constantly get a new piece of tech every couple of months that enterprise engineers are going to ask for. Applications developers are going to say, "Now, I want that. Now, I want that." It's tough to not just go to the cloud, because they're being created there as service products. So that's where we can really come in and we can work with the infrastructure ops folks and say, "Listen, let us help you give the agility and speed back to your internal users, but wherever you want, on your own internal infrastructure, on any of the clouds, and you get the both of worlds. You give your internal teams this public cloud like experience, but you get to decide which open source software you're going to use. You're not going to get locked in to any particular cloud on top of our platform, and then you can run it anywhere."

Sometimes what people say to me is they say, "Well, aren't we just getting locked in to you then if we're running – Are we just getting locked in to Mesosphere if we're running everything with Mesosphere?" The reality is, is because we've made that deliberate decision to pull from the open source community of tools, the Kafkas, Cassandras, as I mentioned earlier, you're not getting locked into us. Your applications – If they're getting locked into anything, your applications are getting locked in to those open-source APIs, and the value we're really providing is we're going to make it easier to operate all those services. If you don't find value in

that, if you really wanted to, you could leave us and you could run all those services yourself, and guess what? Your applications wouldn't have to change. You get to keep running those applications.

So we are giving people the flexibility, and we are we relying on the fact that we can provide some real value in the automation. But it's tricky. I mean, the tension that exists for all these companies is the tension between moving fast, being agile, getting stuff built really quickly and not getting the lock in. I think now is the time to actually help organizations solve that problem. I think you can have your cake and eat it too, I guess is the way to think about it.

[00:33:31] JM: Yeah. I can see how you get a moat there for building really good automation around standing up Kafka, standing up Kubernetes, standing up Storm, Spark, Flink, all these open-source stuff. What when it comes to the high-leverage managed services for which there has not yet been built an open source system, I mean, there are more and more open source alternatives. Like even Knative – I'm just starting to get into that, but it seems like a compelling alternative to building your services on Lambda. But there're probably enterprises out there that are going to want to use AWS Lambda. Then you get into this decision matrix where you have to decide, "Okay. Which open-source services are we going to build and what's the accessibility layer to the cloud providers?" If Google stands up some proprietary magical Google service that we don't have an open source alternative to and we have no idea how to build one, you want to make it available to you users. So you need some kind of control plane where they can reach out to Google or they can reach out to AWS or they can reach out to your open-source managed services.

Basically, I guess the question I'm getting at is you have this gigantic decision matrix of different integrations that you could build. If you're trying to build this multi-cloud magical management layer, how do you handle that decision overload?

[00:34:54] BH: Yeah. Yeah, the conversation that I have with CIOs, CTOs or enterprise architects, the conversation goes like it's something like this, it's very likely that there might be a service on one of the clouds that you want to take advantage of. You mentioned Google. If performance is so critical for the Tensorflow jobs that you're running, there is one place in the world where they have this thing called a TPU for running Tensorflow. If that's the performance

you need, I don't think we're ever going to get TPUs in the rest of our data centers or any other cloud. It's just not going to happen. That's where that's going to be.

Obviously, people will make it work exceptionally well in GPUs, and so they will give you options, but if you get to a place where you feel like there's a service in the cloud, you have to use, I wouldn't say don't do it. What I would say is do it just for that service and give yourself and your company the flexibility for all the other things that you're doing where you can move between clouds or move between on-prem and in the cloud. Give yourself the ability to actually do that for all the other services.

Because if then you do find yourself using a Google specific service and 12 months later you're no longer running that application, or that application runs just as well consuming GPU's in a cheaper way on-prem or in some other cloud, rather than having to do 100% of work to completely move yourself from one cloud to another cloud, you're doing like 10%. So there's some work. You still need to do, because you did tie into some particular APIs. But for everything else, you're just using the open-source stuff.

Again, we've got customers that we work with where they'll use our Kafka, they'll use our Spark, they'll use our Kubernetes, they'll use our Elastic and they'll use then one service from a particular cloud that they want to use that's critical for them to use. But they've still given themselves a lot of leverage and a lot of opportunity in just as much simpler time if they ever want to move in the future to be able to do that move. If they hadn't done that, if they would've built their applications directly against all of those services in the cloud, that would have been tough. Any think about move that they wanted to make, it would've been a very, very long effort. It's unclear that they would ever been able to do that effort. It's unlikely that they would.

But we have customers that have moved apps 100% from on-prem to the cloud, because they've been running on top of the platform. That's powerful. It's really, really, really powerful. So it's very, very possible. Yes, you to move data. Data has gravity. That can be a real thing, but people can do it. They can do it, and I personally think that in the future as multi-cloud becomes more and more a thing, there'll price wars. I mean, there's price wars, but I think you'll see this be a real thing when customers are able to actually apply some leverage and say, "Hey, listen. I

can get it cheaper over here." So I'm going to do that. I think that's a good thing. I think that's a good thing for business.

[00:38:06] JM: So if I want to interact with an Amazon service today and I want to be – Like an Amazon specific service, like Amazon Kinesis, for example, and I want to have my Kubernetes cluster interact with it. I've got some application that wants to make calls out to the Kinesis API, but I want Mesosphere to be my multi-cloud management layer. So I've got certain Kinesis calls that I want to make in my application. I've got certain Google BigQuery calls I want to make. If I'm using Mesosphere as my platform, do I also need like an instance in Amazon? Do I also need an instance in Google, or do I just make calls out to the API? How does that look?

[00:38:52] BH: Yeah. You can you can just make calls out directly to the APIs. There's kind of two parts to this magic. The first part of the magic is how you get the services themselves. How you get instances of the services, because even with the Kinesis to the big tables of the world, you still have to hit APIs to give yourself to provision the service so that you can then make API calls to it. That's kind of the first part. The direction that we see a lot of folks going in and that we're also investing in much around things like open service broker to allow a standardized way of letting some people come in and say, "Oh, I need X, Y, Z. Is that part of my catalog? Okay. Yes, it is. Great. I want that thing. Let's get it provision under the covers," and that I think helps infrastructure ops teams as well start to think about the interface, the catalog if you will, the multi-cloud and beyond catalog for teams. That's kind of one side of it. Then there's the second side of it, which is, "Okay, now once I've got the services provisioned that I just want to hit the APIs, what can my apps do?"

In that case, the apps just talk to the services directly. They just hit it those APIs directly, and that's where some of the lock-in can come in. If those applications are using APIs of these services, which are cloud-specific, or service-specific, that's where it's going to get harder for people to over time figure out how to change the application and have it use some other API, an open API or something else.

Again, I mean, I think the reality is, is sure, it's always software and someone can always show up and they can rewrite the application to talk to this new API, but software is still far more brutal than people realize. Once you make something work to go in then totally rewrite parts of it to

speak is slightly different API, you're going to have bugs. You're going to run into bugs. That's going to lead to frustration from your end users, because they're going to experience those bugs. Then on top of that, a lot of engineers, they don't want to do that work. That's not fun work to go and take an application and change the API.

You don't have a lot of wind in your sail I suppose when it comes to in the future being able to re-factor and really change software. A lot of engineers don't like to refactor software like that. A lot of engineers like to refactor, but they want to refactor and build the blinky version of copy and paste or something. They love to do that, but they don't like to refactor where it's like, "Oh, I'm just swapping it from using Kinesis to Kafka," or something like that, like, "Oh, gees! That's a total drag."

But yeah, in our platform, you wouldn't need necessarily to be running any specific Mesosphere nodes in all of the clouds.

[00:41:34] JM: Okay. You don't need access to like the AWS CLI or the Google Cloud CII or –

[00:41:40] BH: No. No. Yeah. No. If you want to run our services, the ones from the Kafkas and the Cassandras and those things on top of us across multiple clouds, which we have customers doing, to do that, yeah, you're going to need to run us in each of the clouds. We've got some really, really interesting work. We're doing customers, where they'll do exactly that. They have these single massively distributed clusters were we've got nodes in Google and Azure and AWS in multiple regions and then using basically constraints. They can launch these different distributed services in these various regions and they can bring up one for each of the different clouds. It's very, very powerful. It's really, really cool, and they might want to do that, because, again, they're consuming a particular service from the cloud provider, and so they want to place it in that cloud provider's region so that they get better performance from a latency perspective, but they don't have to. They don't have to.

If you want, there's plenty of apps out there that are hitting cloud SaaS services from their on-prem data centers and just dealing with hundred millisecond plus latencies, because they can, or they've been architected to able to do so.

[00:42:52] JM: Right. So as you're building all these integrations with, you got this end-by-end matrix of cloud providers and open source technologies and you need to be able to have an on-prem set up. So you need to be able to – Wherever Mesossphere is deployed, if it's deployed on-prem or if it's deployed on Azure, if it's deployed on Google cloud, you need to be able to provision Hadoop clusters, Kafka clusters, Cassandra clusters. What are some engineering difficulties in building that consistent matrix of distributed systems that could potentially be deployed? Also, how do you hire enough engineers to build all those integrations?

[00:43:37] BH: Yeah. I mean, the first step for each of these each services is having enough expertise to be able to understand what it takes to operate them so that we can build in the automation via software of these services. That's kind of step one. For that, the folks we hire are people that have run these things themselves. There's a great term for this, and I'm just blanking at it, but in anger. That's what it is. We like to hire people that have run these things in anger, because they've experienced the pain. When they've run these things in anger themselves, I'm sure that they've either written a lot of code themselves or scripts or whatever it is to try to operate these things themselves, or they've at least written down all the things they might want to do. So we'd like to bring those people in and say, "Great, let's turn all that into code. Let's turn all that into code that actually operates these things, makes it easier to operate these things."

The question I often get is – The question I often get is, "Why don't the projects do this themselves? Why don't the open source projects do this themselves? If these things can be such a pain in the butt to operate, why don't they just do this?" But the reality is, is that they don't do it because they're run in enough disparate ways that it's not easy to just be like, "Everything will be deployed in this one way, and so therefore we can make the operations of this thing perfect."

[00:44:59] JM: So luckily, that's falls on your shoulders.

[00:45:00] BH: There you go. Exactly. So luckily we get to take that on. But that's kind of step one. Then step two is, okay, which I think is one of the most important steps, is how do all these things run with one another? I think that that's so important, because if you really want to do things in a cost-effective way either on-prem or in the cloud, these days you got to run the stuff

together. Because you've got to take advantage of vital resources, you go to drive up your utilization.

When we first started this whole container scheduling resource management thing, I mean, we don't use the word resource management anymore. It's not the word we use, where at KubeCon we talk about container orchestration. It's about orchestrating your containers for more than it is about being efficient with resources. You're getting that. That's a part of it, but this is less of a big part of the conversation that we're actually having.

But when it comes to bunch these distributed systems, it's so important, it's so relevant, because if you really wanted want to drive your cost down, you need to run these things together. So that's kind of the second part, is once you figured how to deal with the automated operations of these things, then it becomes how do you deal with the running these things alongside other things? What happens when Spark and Kafka run on the same node? What happens when Kafka and Cassandra run in the same node? When happens when you run too many Kafkas on the same nod? What are the performance things that you run into? What are the other weird things that happened because these two services both happen to use disks in a particular way? That's the whole second phase of the automating the operations that we like to, again, say, "A lot of people haven't done that in anger, because they didn't run them on the same node. But if they wanted to do things in a cost-effective way, they're going to really have to." That's the reality.

I mean, when you go to AWS and you get services, all these stuff is just – It's done behind the scenes, and I mean there's no question in my mind. Well, I've never worked at AWS. There's no question in my mind that there's engineers at AWS that are building exactly this kind of software for their SaaS offerings, for their services that they expose, and that's of course not open source at all. That's 100% proprietary, and you pay for it. The margins for those services at Amazon that, they're probably huge.

I mean, we've heard Amazon's margins are 30s or something like that, 30, 40, or some like that, but the margins for Rackspace for using VMWare was like 10 or something like that. So let's say AWS is doing a little bit better than them. Okay. So they're doing a little bit better than

Rackspace. That means that the margins for the services have got to be huge, 50, 60, huge, massive, to actually build account for the number of VMs running versus everything else.

Yeah, there's real value in that software. There's real value in giving you that as a service experience, because there's a lot of thought and thinking that goes into it. So, I guess to answer your question. Yeah, we hire folks that have done it in anger and we turn that into software that helps manage both the day zero deployment install as well as the day two upgrades, Elastic scale up, scale down, all those pieces. Then we also put a lot of time and effort in to run all these things together, and let's see where stuff falls over, and when it falls over, let's figure out why and then let's figure out what we can do to actually improve these pieces so you can run them together, because that's how we're going to drive up the utilization across the clusters. Drive down the cost and give those companies a far cheaper experience than if they were to just spin up each of those services on any of the public clouds themselves.

[SPONSOR MESSAGE]

[00:48:50] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM.

Now Datadog has application performance monitoring for Java. Start monitoring your microservices today with a free trial, and as a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/data.

Thank you, Datadog.

[INTERVIEW CONTINUED]

[00:49:44] JM: So when you were at the AMPLab, your work on Mesos was building a two-layer scheduler around distributed systems, and the scheduler problems that I've talked to Ion about,

for example, he's thinking about serverless right now and the idea that you can make calls out to serverless functions and have them be provisioned easily and they can be performant and perhaps you can do interesting things on top of those serverless functions, like do MapReduce or distributed machine learning atop a set of Lambda functions or serverless Knative things or whatever your function as a service unit of compute is.

Given your expertise in schedulers, what do you think of that space? What are the fundamental problems around building functions as a service platforms and building the level on top of it to be able to do higher level service construction?

[00:50:51] BH: It's interesting. Functions as a service, AWS Lambda and all the other ones that are coming out, when they first came out they sort of – I think they took people – It's like massive hype initially. But then I think it's been a little bit quieter. I mean, it's not that quiet, let's be honest, but just compared to say some of the container orchestration stuff that was happening. Getting back to the earlier part of the show, I think one of the reasons for that was that functions felt a little more revolutionary than evolutionary. Containers feel very evolutionary for people. Even if there're some revolutionary things there, I think it's just far easier for a lot of enterprise shops to think about how they can start to adopt containers along better their evolutionary PaaS that they already have at their companies.

In fact, I can't tell you how many companies out there that we've talked to, they literally just treating the container like a VM. They're sticking all the same amount of stuff that they would have stuck in a VM in a container and then they're just running it, which is really unfortunate, because then when they go to run these 3.6 gigabyte containers and it takes five minutes to start the container, they're asking themselves, "What the heck? This isn't any faster than VMs. What what's going on? I thought that this was supposed to be superfast?" You're like, "Well, it turns out you've got a lot of crap in that container, because you're treating it like a VM." Anyway, that's a bit of a digression.

So I think one of the reasons why –

[00:52:18] JM: Well, not completely, because people are doing that with functions.

[00:52:21] BH: They are, yeah. They are, yeah. But to me, what's so interesting about functions is when you think about how you build software, it's just a collection of functions. That's what it is. When you take a step back and you think about, “Okay, I'm building this new complicated distributed system, whatever it is,” it’s like, “Oh, I’ve got this functionality, I’ve got that functionality,” and it’s just going to be a bunch of functions that things call to, but the way in which we really deploy these then distributed systems is we have higher-level concepts of components that are collection of a whole bunch of these functions, and then we deploy basically the components and we manage the components.

Functions as a service is pushing that to an extreme and it's basically saying, “Stop thinking about the components, and just think about the individual functions that you're actually trying to provide.” So I think where there's still open questions in my mind and I think in a bunch people's minds is, does pushing it to the limit of just thinking about functions, does that end up getting – Does it make it harder to build certain things? Is there an impedance mismatch with certain kinds of software they you’re trying to build?

You mentioned things like people are thinking about they could do MapReduce on top of functions. They could do this on top of functions. They could do that. I think you can. Yeah, 100%, because all software is really at the end of the day just a collection of functions. We’ve just chosen arbitrarily to group those functions into components that we then deploy the components and containers throughout our clusters.

So I think what will be interesting is as people are going to kind of push functions to this limit, what are we going to be doing from a software engineering and programming languages perspective to make some of that stuff easier, or is it just going to kind of to be this clogy mess of actually there is an impedance to mismatch for this kind of thing I'm trying to build, and I'm going to bundle a whole bunch of stuff in my function, as you alluded to earlier that people are already doing, or just do really weird stuff in my function, because it's not exactly what I want. But there's other aspects of functions that I like and it’s super easy to deploy these functions and everything is really managed for me, because I'm just launching these functions. So I’ll make it work.

One of the examples I see is people, they will – When their function gets turned off, they'll fire an alert to restart a function. Like do things are kind of going around the whole concept of this function, like, “Hey, I only need the functions up when there's APIs calls coming in, and when there's not, I’ll shut them down.” Then people are like, “Oh! But for this thing I'm trying to build on top of functions, actually, the stuff in memory is really, really important for me. So if my function gets shut down, I need to start it up right away and pull stuff into memory right away, because it's going to be a lot faster, because that's how I do a faster MapReduce,” or something like that with functions, and they’re kind of going around the intent of how function should be done. To me, that's kind of an impedance mismatch. It's not a one-to-one with how function should actually be run.

But I think it's still early days, quite frankly, and that's why I think somebody like Ion at Berkeley to be wanting to take this stuff on and look at it is to be able to take those kinds of impedance mismatches that people run into and ask, “Can we put some new abstractions or some new primitives as part of functions to actually make this work really, really well and capture the kinds of programming models or programming tasks that people are trying to do while not losing sight of the really, really nice property of, “I just got to deploy a function. I just got to send a function out there.”

I think it's a pretty liberating programming model personally. When we built Mesos, we build Mesos with the actor programming model. Everything is asynchronous. So we've got a ton of “functions” in the platform. Really, you can think of it as a ton of actors running around sending messages to one another. I think that's a really liberating programming model to be able to think about all these independent things that are just running, and when they fail, they restart. When they want to talk to somebody else, they just send an API call and then they can wait asynchronously for the response, or not. They just broadcast it out there and then that's it.

I think from a programming perspective, there’s a lot of power to it. Again, the word I use was liberating, because a lot of our program models, they’re so intertwined and like your software you end up producing is so complicated and interwoven. I think the functions as a service can help get people more into the perspective of let's think asynchronously and a bit more independently about a bunch of independent things that all have APIs that they expose or that

react to events. Then together, all the work that these things are doing, make up this grander, bigger picture, this application.

But at the same time, as much as I think that's liberating, I think it's still early days and I think as people push the limits of how they want to run certain software on top of it, I think the natural thing that they're going to run into is like, “Oh, this doesn’t exactly do what I want to do. So I’ll do this clugy think to try to make it work.”

I think we will overcome that. I actually think that functions will be a much bigger part of our future than people realize, possibly even sooner than we think. It will be interesting to see how that plays in with the container world. Container orchestration containers, they've already really been commoditized with the help of Kubernetes and a bunch of other services. Commoditized in a sense that people will do them and they'll use them in a sort of a standards way. But I think that people might jump even faster to functions if we can make it super easy for them to be able to consume those functions as they’re thinking about rebuilding their applications versus sticking those applications in containers.

I will say though that I think it's going to be a while until we’re building really sophisticated distributed systems like the Kafkas and the Cassandras and all these things on straight functions, just because I think that there’ll be enough impedance mismatches that people will have that they'll basically say, “No. You know what? I just am going to still build this in the “traditional way”, the bundled software where here’s all the software and I go.”

Even distributed systems, I think that's been tough for people. Like bundled software was easy. I stuck a CD in my computer, I double-clicked the install manager, it installed. I was done. Distributed systems are still far more complicated.

[00:58:54] JM: Even with Help charts?

[00:58:55] BH: Even with Help charts, because it comes to the operations of it. It comes to like all the other things. I still think for – So somebody that wants to build one of these really complicated distributed systems and go just have their deployment mechanism be functions. It’s 10 years ago, my distribution mechanism was here's a CD. Five years ago, my distribution

method was download this tarball or this ZIP file from the internet. Now, my distribution method is looking – Is getting like a bundle of containers, and I think that's still messy. I think it's still messy even with Helm charts, even with those things, and that's why I think a lot of people are going to services. To get back to earlier conversation earlier, I think that's why they just want to go click a button and get those services, because there's still just too many piece under the covers.

So I think like how are you going to distribute the functions of software? You're probably not. If you're exposing some software you built with a bunch of functions and you are running it by a functions, that's one thing. That's totally one thing. But if you're trying to ship software to somebody else, you're trying to build some distributed system the you want other people to be able to run on their own, I think we're still a little ways out there until someone says, "Here's my new system. Before you run it, you have to install Kubernetes, then you have to get Knative going, and then you can launch my functions on top of Knative." It's a lot of steps.

[01:00:15] JM: A lot of steps.

[01:00:16] BH: I just don't think that you're going to see that next year. Maybe you'll see it next year. You never know these days in the industry. But I see more. I see functions are going to – I think they're going to be used by a lot by a lot of teams, and they might use those to then ultimately provide some service out to other people. But I don't know what it's going to look like from a distribution of the end software.

Again, as we're moving to this world of everything is being operated, consumed and operated in as a service way, maybe it doesn't matter. Maybe in the next five years, we won't be distributing software. Everybody will just be consuming everything as a service.

[01:00:54] JM: That's what I see today in younger companies.

[01:00:56] BH: With younger, it's 100%. Yeah, that whole cloud 2.0, cloud 1.0.

[01:01:00] JM: They move so much faster. It's all APIs and firebase and Netlify and Heroku and just like – They just want to move fast. Can do business logic and –

[01:01:10] BH: That's exactly right. You got it. Yeah. Again, I think that's where the traditional enterprise companies get stuck and they're like, "What do I do?" And that's where we try to come in and say, "Well, let us help you give your internal customers a bunch of the services to move fast."

[01:01:24] JM: They need to hurry up.

[01:01:25] BH: They got to hurry up, because otherwise they're all going to go to the cloud.

[01:01:27] JM: I'm looking at the pace of these companies that are doing cloud 2.0. They just move so much faster.

[01:01:34] BH: They totally do. Yeah.

[01:01:35] JM: And they're happier, by the way.

[01:01:37] BH: That's one of the funny things as well, is you look back at the Linux cons of 10 years ago, a big part of that community was people running all the tech themselves, like the operators, the administrators. That's what they got to do. That was like a big part of it. Now you look at like the brand-new startup in Silicon Valley that just got funded and they're just straight using the services and they're like, "What? I don't know how to manage a MySQL database. Why should I know how to manage a MySQL database?" "Well, didn't you get a computer science degree in college?" "Of course, I got a computer science degree, but whatever. I'm building these higher level machine learning applications or I'm building these other applications. Just because I got a computer science degree doesn't mean that I should know how to run my own MySQL database." Yeah, that's that cloud 2.0 mental mindset that I think five years from now enterprises will look back and they'll be doing that in some capacity. We'll be helping them to do that.

[01:02:32] JM: I mean, so much of this conversation has been premised on this idea that – And so much of your product, by the way, is premised on the idea that these enterprises want to have an avoidance of vendor lock-in. Sometimes I look at it and I'm like, "Your business is so

good. Why do you care if you're spending $1 million or $10 million managing your servers and your infrastructure? Just like pay AWS to manage all of it, or pay Google Cloud to manage all of it, or do whatever you need to run faster, because otherwise your core business might actually be threatened. Either way, it's a de minimis percentage of your budget relative to what you – I mean, does it ever just seem like complete madness to you?

[01:03:17] BH: I mean, for companies that are just like, “Wow! We cannot move at all, whatsoever. We are moving so slow. What do we actually do?” I think it's easy to just look at the cloud and say that’s a panacea. That's going to help my companies move faster. That’s going to do everything else. But I think enough people of been through that experience with other –

[01:03:37] JM: The lock-in.

[01:03:37] BH: Vendors. Yeah. They've got the gray hairs to deal with the fact that they were writing hundred million dollar checks to –

[01:03:46] JM: So like the business equivalents of the Ion Soica’s endemic experiences.

[01:03:51] BH: That’s exactly right.

[01:03:52] JM: They know.

[01:03:52] BH: They know.

[01:03:53] JM: They’ve seen this rodeo before.

[01:03:54] BH: You got it, and they’re sitting there and they're just like, “Whoa! Okay. Hold on a second. I've already got these huge checks that I'm writing. Is there anything I can do?” I mean, they’re at least asking the question, which I think is the right thing to do. Is there anything I can do to protect myself? To protect the business? But yet at the same time they’re like, “But I want to move fast. I want to move fast.”

The charge that they have for their infrastructure IT organizations is how can we move fast while giving us the flexibility in the future? That's why I think it's – That's why we exist as a company, and that's why I think it's a really important time for us to be working with all those organizations, because I think you can get your cake and eat it too. I think that you can get the agility and the flexibility and provide things like message queues as a service without getting locked into a single cloud, and I think it's just the beginning of that. Because, really, three years ago we didn't talk that much about the three clouds. We still just kind of talked about AWS. Now we're talking at least about two clouds for sure, and we're talking about more and more other clouds as well.

So we think you can have your cake and eat it too, and we also think that when you're going to start deploying your software in a bunch of more interesting places, like the edge, that's the other big thing, is running your software at the edge. You're going to want this flexibility. You're not going to want to find yourself. All your applications can only run in the cloud and you want to run your application in some really interesting environments for whatever business you're trying to do and you don't want to ship data and deal with the latencies and all the bandwidth issues and everything else. What are you going to do? What are you going to do? We think we're going to see the edge more and more, and that's just going to make this multi-cloud and this hybrid cloud thing be a reality for organizations.

[01:05:34] JM: Okay, last question. Since you brought up edge computing, AWS announced this out Outposts thing, and Azure has Azure Stack. So the cloud providers movingly into the on-prem world, in your conversations with enterprises, do they – What's their level of interest in this? How does it impact your go-to-market strategy?

[01:05:59] BH: Yeah. So Azure kind of kicked it off, Azure Stack. I don't know that it's been a huge success. We don't see a ton of it, honestly, with a lot of the customers we work with. We work with folks who are on Azure. They're not on Azure Stack. They're using Azure in the cloud. They're not using Azure Stack.

Google started at least with Kubernetes GKE on-prem. I'm pretty sure GKE on VMWare on-prem as something that they've started. I don't know actually where that's gone. So I think for AWS to announce something, it's almost like they had to. I think they had – If they didn't announce it, I think there was just going to be far too much, "Hey, we're thinking about going

with Azure, because they're going to give us an on-prem experience, or we're going to go with Google, because they're at least starting to think about on-prem and we know we have to have some on-prem."

The way it's been described, and I don't work at AWS, nor have I had a deep conversation with AWS, but the way it's been described is it's manage. It's managed on-prem, which I think is going to be a turnoff for a lot of fun enterprise organizations. A lot of enterprise organizations, the infrastructure folks, they still want to get the software. They still want to run the software themselves and there's very –

[01:07:04] JM: And they want to keep the date on-prem.

[01:07:06] BH: They want to keep the data on-prem. So when I say managed, I just mean there's going to be an AWS employee who is operating your clusters. So I do think for some organizations that's going to be like, "Yeah, it's fine. I'll bring these people in for other organizations." You're going to be like, "Whoa! Whoa! Whoa! That's not the model we want. We want to be able to run the AWS software in ourselves."

So from that perspective, I mean, again that comes back to what we're trying to do is we are trying to enable the infrastructure operators to better do it themselves, but still make it a lot easier, as easy as a public cloud. I think that's still an opportunity for us to do that with those customers.

The second thing is our focus on the open source services. I still think that even if you go on-prem, a lot of these organizations are going to want to use the open source services and the open source APIs to give them that ability to move if they want to. To still not get locked in to any particular provider. So I think that that's to be a thing.

Again, you have AWS announcing things like the managed Kafka service at AWS Reinvent. So I think they're all securing customers saying, "Yeah, we want more open source APIs. We want more open source APIs," but I don't know that they're going to push that to the extreme the way we push it to an extreme, which I still think is going to be some opportunities that we have for a bunch of the customers that we work with.

The last thing is I don't know that – I'm not convinced that the future is just going to be a hybrid world. Hybrid being you pick one public cloud provider and you pick, and they can also provide an on-prem experience. I think it's going to be more mixed. I think the future is going to be a multi-cloud world. The reason why I think the future is going to be a multi-cloud world is because as part of the conversation we had earlier where there's going to be some things you can do on particular clouds that is better than anywhere else. It's better than any on-prem experience you'll ever get, or it's better than any of the other public clouds.

[01:08:57] JM: Then there's lots of vendors that are cloud agnostic. Your Cockroach labs, or your Stripes or –

[01:09:03] BH: Yeah. So the same thing that I said earlier, is, well, then if you go to run stuff, consuming that particular public cloud service, everything else you should run in that public cloud you should run it in a cloud agnostic way where you can be able to move it if it any point in time you don't want to use that service, which means I just think that companies that are being thoughtful about this from day one, they're going to decide not to just – Even though they can run AWS on-prem, they're still going to decide to do it in a multi-cloud way. I think the whole like, "Oh, I can run this stuff on-prem," becomes less interesting for a bunch of those companies.

It's not to say that I don't think people will give it a shot and some people will dive in. I mean, it's a huge market. It's a massive market, but I think the real opportunity for us still is, "Hey, we give you multi-cloud, we give you on-prem. It's an open ecosystem of services, open source APIs. We give you that public cloud experience though on-prem and any of the clouds, and that I think is still going to be a value proposition for a bunch organizations."

[01:10:05] JM: Ben Hindman, always a pleasure.

[01:10:06] BH: Yeah, thanks so much. Great to be here.

[END OF INTERVIEW]

[01:10:11] 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]