EPISODE 584

[INTRODUCTION]

[0:00:00.3] JM: Mesos is a framework for managing distributed systems. The goal of Mesos is to help engineers orchestrate resources among multi-node applications, like Spark or Hadoop. Mesos can also manage lower-level schedulers like Kubernetes. A common misconception is that Meos aims to solve the exact same problem as Kubernetes, but Mesos is a higher level abstraction.

Ben Hindman cofounded Mesosphere to bring the Mesos project to market. Large enterprises like Uber, Netflix and Yelp use Mesosphere for resource management. Before he started the company, Ben worked in the Berkeley AMPLab, a research program where the Spark and Tachyon projects were also born. At this point, Ben has spent significant time in both academia and industry, and this conversation spans distributed systems theory, practice and history.

Ben and I spoke at Cube-Con 2018 in Copenhagen, which was an amazing conference. We were both amazed at how big the audience for Kubernetes has gotten and the pace at which the technology is advancing. Today, Kubernetes is mostly used for scheduling containerized applications that engineers have built themselves, but there will be higher-level tools that use Kubernetes as a building block, much like Zookeeper was used as a building block for Hadoop, Kubernetes will be used to build serverless applications and distributed databases.

Once you are using a distributed database built on Kubernetes, you may not want to think about the container orchestration that's going on. You want to think about the raw storage and CPU requirements for that distributed database. This is one reason why Mesos compelling. Since Kubernetes creates an increased cardinality of distributed systems, it's good to know that there is a framework like Mesos built to manage those higher level applications.

I really enjoyed this conversation with Ben Hindman and I think you will too. Let's get on with the episode.

[SPONSOR MESSAGE]

[0:02:17.8] JM: Today's podcast is sponsored by Datadog, a cloud scale monitoring platform for infrastructure and applications. In Datadog's new container orchestration report, Kubernetes holds a 41% share of Docker environments, a number that's rising fast. As more companies adopt containers and turn to Kubernetes to manage those containers, they need a comprehensive monitoring platform that is built for dynamic, modern infrastructure.

Datadog integrates seamlessly with more than 200 technologies, including Kubernetes and Docker so that you can monitor your entire container infrastructure in one place. With Datadog's new live container view, you can see every container's health, resource consumption and running processes in real time. See for yourself by starting a free trial and get a free Datadog t-shirt at softwareengineeringdaily.com/datadog. That's software engineering daily.com/datadog.

Thank you, Datadog.

[INTERVIEW]

[0:03:26.4] JM: Ben Hindman, you are the chief product officer at Mesosphere. Welcome to Software Engineering Daily.

[0:03:30.8] BH: Thanks, Jeff. Great to be here.

[0:03:32.3] JM: Yes. We last spoke several years ago on Software Engineering Radio, and that was closer to the time when you had been working at AMPLab in Berkeley. I think you would just started Mesosphere when we last spoke. How has the field of distributed systems changed since you were working at AMPLab seven years ago?

[0:03:54.3] BH: Well, that's a great question. I think there's been a couple of things that from my perspective now being outside of the AMPLab and in the industry that I think are interesting to think about. One is distributed systems are far more commonplace in the last couple of years than 2009, 2010 when I first came out of the AMPLab. I think a lot of people used to think that you needed to have a Ph.D. to be building and operating or working on a distributed system. It was usually kind of the folks that were working on that stuff. It was those with the people that were really doing the work.

Now I would say whether or not people realize it or not, everyone is pretty much working on the distributed system. They're either interfacing with some other API, in which case they need to think about what happens if there's any kind of issues or failures with communicating with that service, or they actually are building themselves multiple different components that are all communicating in some kind of a distributed way.

I mean, it's sort of — It was interesting to actually — I got my bachelor's degree at University of Washington and when I was at University of Washington and the whole sort of big data Hadoop movement was happening, all of a sudden the big data classes started getting created at certain universities and University of Washington was one of the first ones that had it and I know that because one of the founders of Cloudera actually came in and help run that course.

We always had distributed systems courses, but I started to notice even at Berkeley that there was now distributed courses not just being taught at the master's or Ph.D. levels, but even at the undergraduate level.

Anyway, I think that's probably one of the biggest shifts that I've seen, is it's just much more commonplace now. Pretty much everybody is working on, or doing a distributed system and I think that's great, because that's sort of is the requirement these days when you think about the kind of applications that we want to build. They need to be distributed in some nature. So I think that's a good shift.

[0:05:51.8] JM: Has there been a complete convergence in the questions that theoretical institutions are asking and the questions that industrial institutions are asking?

[0:06:03.1] BH: That's a good question. I don't think entirely. I think the other thing that really has happened is that certain hard problems of distributed systems have gotten solved and sufficiently abstracted away that now people can just rely on those things. So a great example is going into my time at Berkeley, if you wanted to talk about consensus, hard distributed systems problem, you pretty much had to read the Paxos paper, and most people can't get through that thing.

So if you then looked at the kinds of distributed systems that were coming out roughly around that time and even a little bit later, you had people that were trying to build systems to solve that problem so that other people could take advantage of the system. So, I mean, you always had distributed databases and all the work around two-phase commit and everything that was going on there. But now what you started to have is you sort of have things like Zookeeper, and this was a significant system to, again, sort of — I don't know, commoditize or democratize people being able to do really complicated other distributed systems work, and that was a big shift. Now, what I think is really interesting about that though is things like reconfigurable distributed consensus. If you think about Paxos, the way in which everyone had initially built Paxos is it was like these three nodes are these five nodes that they were going to be doing consensus forever, which is just not realistic, because eventually one of those machines is going to die or maybe you want to go from 3 to 5 nodes, and so actual work on reconfigurable Paxos was like in the 2,000ths. Even if Paxos was way, way earlier, and we'd finally gotten to now building more robust systems like, say, Zookeeper, things like reconfigurable Paxos is not that old.

When you ask your question about sort of what's happening from the academic space and then versus what the practitioners are doing, it was still a lot of really interesting work that is critical, that was still going on. The other thing that was really happening was even Zookeeper, while it shared a lot of the same ideas from Paxos, it had its own consensus algorithms. It was Zab, if I remember correctly.

[0:08:28.7] JM: Zookeeper atomic broadcast.

[0:08:29.7] BH: Exactly. Yeah. So then they were doing their own reconfiguration, which came out again — I don't know. I want to guess like 2012, 2013, maybe 2014, right? So they were answering their own sort of similar questions which had only recently been answered for Paxos, and then of course you had Raft, right? Which was now basically people taking a step back and being like, "All right. Hold on a second. Now we've had some time of actually trying to implement Paxos in practice of coming up with new variants of consensus, like Zab. Is there a better way to think about this, a better way to do it?"

So you definitely had a convergence at least in the industrial side and the academic side of, “Okay. It's clear that we need these systems to be built so that they can be a foundation for all these other distributed systems work they were doing.” So you saw that convergence, but I think that just sort of led to, now, people thinking even harder in the academic space. Can we do this even more efficiently? Is there completely new algorithms we want to do?

I mean, technically, Raft came out of academia and it was implemented rather quickly, and then used in a pretty widespread manner. Again, there still a lot of people that I know that are still trying to think if they can push the boundaries even farther. I mean, just I think last year there's still papers around things like improving Paxos performance with batching and pipelining and bunch of other things.

I definitely think that, if anything, the fact that we wanted to use all these stuff in the real world has motivated again some work in academia, which I think is a great thing. Has it completely converged? They’re at least tracking in a much more similar direction than they were 30 years before, which things like Paxos got created and a few very small people did something around and then all of a sudden age of enlightenment for consensus and [inaudible 0:10:23.5] systems and everyone is doing stuff around it.

[0:10:26.0] JM: Then you have the cryptocurrency people, often their own third area, the age of bacchanalia perhaps.

[0:10:33.1] BH: Totally.

[0:10:34.0] JM: Something else.

[0:10:34.7] BH: Yeah. I mean, that’s exactly right, and that in itself I think will change the way we think about doing distributed systems.

[0:10:42.4] JM: It's like the 60s of distributed systems. It's like they've all taken crazy drugs and trying new things and letting their flag fly. But Zookeeper, I think, is analogous to Kubernetes in the sense that it was a primitive that unlocked Kafka and unlocked Storm, I think?

Obviously, Hadoop. Then you see the bellwether, the same kind of thing happening with Kubernetes. It's obvious that there are going to be some — Oh! You need to unlock your iPhone in order to access Kubernetes?

[0:11:17.5] BH: Yeah, I guess that's what happens when you think — Someone of us said Siri. One of us said Siri. I'm not sure how, but one of us said Siri.

[0:11:27.2] JM: So what's going to be built on top of Kubernetes if we think about the analogy to the Zookeeper?

[0:11:31.9] BH: Yeah. It's interesting, because when you think about Zookeeper, there's etcd which literally ended up being in the Zookeeper for the Kubernetes world, and it's kind of like Spark and Hadoop. You had Hadoop and you had Zookeeper in that whole world, and then you sort of had this evolution. Eventually people said, "Okay. We're not going to do Hadoop map reduce. We're going to move to Spark," and I think the same thing happen with Zookeeper. People ultimately said, "All right. Could there be something else?" Which I think is pretty typical that [inaudible 0:12:03.8] some software built and then they'll be another version of software that people will end up consuming.

Etcd has been much more popular than just for — Etcd and both Zookeeper, as you're mentioning, they have unlocked the Kafkas and all those things, but I think they've also just changed the way a lot of organizations have thought about even things like doing service discovery and how they want to do — There's just been a lot of other really, really interesting applications of it.

What's going to be the next stuff on top of Kubernetes? It's interesting. Here, we're at Cube-Con right now. I hope that's okay to mention. Yesterday you heard in a couple of the different keynotes, people sort of asking a similar question and wondering in particular whether or not there'll be a convergence of serverless and containers and whether or not that will sort of be, if you will, one of the next big things on Kubernetes. I think it's a really interesting question, because there's two ways to approach that and there's two — I will give my prediction, but I could be completely wrong.

There's one way to approach that, which is to take Kubernetes and have it evolve into something which you can use in both a serverless function as a service kind of way, and then there's another path which would be to take something like an open [inaudible 0:13:29.3] or one of these solutions and just make them be incredibly successful on top of Kubernetes. The reason why I think this is actually really interesting question and gets back to your asking like what's going to be the future things on Kubernetes, is because I actually think it's a real dilemma, because the easiest thing for mass consumption of serverless would be to put it into Kubernetes itself, to make it be just part of the APIs, because then everybody is going to be able to have a one stop shop to be able to go, pull on Kubernetes and be able to run functions.

But the reason why it's real dilemma is because the complexity of bringing all that into Kubernetes itself is nontrivial, and it just adds to the overall complexity of distributed systems. We're talking earlier, we're building distributed systems. They're still complex, even though everyone's doing it. It's still a really, really complex thing.

I mean, I think this is a real dilemma, and you're already seeing in the Kubernetes community, you're already seeing people, maintainers, contributors trying to push back and little bit and say, "Hold on. What do and don't we want to pull into the core here and is there other things we can pull out?" Because you want to get that to be a rock solid piece of software. You want that core thing to be rock solid. Even if you're constantly sort of bringing new things in, and that could threaten the stability of the software, is there a real dilemma?

[0:14:55.3] JM: Is that because if you wanted to poke serverless technology into the core Kubernetes APIs, it's like trying to just shuttle in the most complex scheduler possible.

[0:15:06.0] BH: Yeah. I mean, it's basically just trying to shuttle in all the components that themselves will have a life of their own and evolve into this lower level system, which also has a lot of stuff it needs to do, and it just — Software gets bigger. Maybe they want to go in slightly different directions. It might become harder to easily define some of the interfaces and intersections once everything is kind of within the same code base. In many ways, it's kind of like asking the question, "Do you want to Kubernetes a monolith or more of a micro services architecture?"

We are promoting — A whole part of cloud native is everybody go micro services because they have independent components that can all communicate with one another. Why? Because even though it is way harder to manage and you need things, like container orchestration, resource management to run all your micro services from a building perspective and an isolation perspective, it's way easier to think about. I've got my thing. It does this thing. It doesn't well and it interfaces and communicates to those things, versus the monolith approach, like throw it all on the same thing and let's figure it out.

Anyway, I think that it's going to be interesting thing to see how that involves, because I think one of the things that Kubernetes has historically done really well is being a one stop shop for folks. They can show up and do pretty much everything they want to do, and as serverless continues to grow, it'll be interesting to see if the project decides to pull it in or ultimately if it remains something that runs outside. I can't speak for the maintainers, but if I was maintainer, I'd probably be in the position of let's keep this thing outside, because I have got my users to think about here which are running this and want to make sure that they're not going to run into issues. But on the flipside, it creates a more potentially cumbersome or more difficult experience. You can work around that, but it's just a thing to be considered.

[0:17:15.7] JM: But we're still in the let a thousand flowers bloom stage anyway and there's 5 or 15 or 20 or 50 serverless frameworks built on Kubernetes. Let them sort out the hard problems. Maybe if we come to a consensus on what is the best way to do it, then, okay, pull it into Kubernetes. But until then, let the hackers —

[0:17:38.0] BH: I totally like that approach as well. That's an approach that I would always advocate for and want to do. The toughest part in today's software though is, which everyone emerges, that thing itself, it's highly likely might have twice the number of lines of code as Kubernetes by that point in time.

One of the interesting things is sort of the higher up the stack you go, the technology stack always seems to just be more lines of code. It could just be even harder at that point to decide to converge the two. But it is an interesting question, because I think what's on a lot of folks — Not a lot of folks, but what's definitely — I mean, this whole serverless thing kind of came in with a crazy bang, right? We sort of talk about hype cycles a lot in this industry, and I think you had crazy hype for Docker. Everyone is like, “I don't that we’ve ever seen hype like this.” I mean, you had crazy hype for open stack, right? People were like, “Whoa! We’re at peak hype.” I think you’re at crazy hype for Docker and they’ll be like, “Whoa! We’re at peak hype,” and then with crazy hype for Kubernetes people are like, “Whoa! We’re at peak hype.” I just think as the industry continues to grow and as more and more people need technology, I just think — I don’t think we’re ever going to get at peak hype. I think we’re just going to — I think, and it could be wrong.

But serverless was kind of this interesting one where it totally peaks, like everybody was talking about serverless, serverless conferences — And a lot of people are doing serverless. Don’t get me wrong. I’m not saying that even though we’ve got the peak hype, there’s not actually the transaction. But I think because there’s enough sort of almost overlap between containers and container orchestration and serverless, it hasn’t been the same sort of like peak hype and then one thing has been forgotten about and the next thing is doing it, right?

A lot of people are like, “Oh, well. Okay. This year we’re doing container orchestration using Kubernetes running containers. Next year we’re just going to be doing serverless.” I'm not sure that's going to happen, but I think it's like it is a question in a lot of people's minds. Is that — Should we stop thinking about containers and should we just start thinking about the functions we want to run?

[SPONSOR MESSAGE]

[0:19:57.8] JM: The Casper Mattress was designed by an in-house team of engineers that spent thousands of hours developing the mattress, and as a software engineer you know what kind of development and dedication it takes to build a great product. The result is an exceptional product. When you put in the amount of work and effort that went into the Casper Mattress, you get something that you would use and recommend to your friends, and you deserve an exceptional night’s rest yourself so that you can continue building great software.

Casper combines supportive memory foams for a sleep surface that's got just the right sink and just the right bounce. Plus, its breathable design sleeps cool to help you regulate your temperature through the night. Stay cool, people. Stay cool.

Buying a Casper mattress is completely risk-free. Casper offers free delivery and free returns with a 100 night home trial. If you don't love it, they'll pick it up and give you a full refund. Like many of the software services that we have covered on Software Engineering Daily, they are great with refunds.

Casper understands the importance of truly sleeping on a mattress before you commit, especially considering that you're going to spend a third of your life on that mattress. Amazon and Google reviews consistently rank Casper as a favorite mattress, so try it out. Get a good night's rest and outvote it yourself today.

There's a special offer to Software Engineering Daily listeners. Get $50 towards select mattress purchases by visiting casper.com/sedaily and using the code SEDAILY at checkout. You'll get the select mattress purchases If you go to casper.com/sedaily and enter the code SEDAILY at checkout.

Thank you, Casper.

[INTERVIEW CONTINUED]

[0:22:03.7] JM: Certainly a lot more money in the containerization Kubernetes migration vendor sales space than there is the serverless migration space as far as I can tell.

[0:22:15.8] BH: I think there definitely is. Yeah. I mean, you have your big cloud players who have serverless offerings, but that's it. You've got —

[0:22:24.0] JM: But vendors don't even know how to — Enterprises don't even know how to digest that stuff, right?

[0:22:28.5] BH: Most don't. Most are still trying to figure out this transition between VM's and containers, and I think they're not even at the point yet of digesting functions as a service exactly as you just said.

I think one of the reasons for that as well is because we're also going through the sort of interesting transition where functions as a service are closer to PaaS in many ways than containers are from the perspective of the functions as a service idea is much more like, "All right. I've just got my language runtime, the libraries I need and now here we go. Here's my functions, or functions that I'm going to run. "Hey, system, you figure out how to get this thing up and running and go," which is much more closer to the whole PaaS world then containers. Containers were sort of like, if people kind of — PaaS didn't digest well with them. They're like — Speaking as someone who entered into the whole container space a while back and everyone was like, "Why aren't you just jumping on the PaaS bandwagon?" For me, "PaaS didn't digest well." "What" That way too opinionated. Do you want me to force me to do something in a very particular way? Just give me some resources and some CPUs and some memory and let me just run some — Whatever software I want to run.

[0:23:42.4] JM: But Cloud Foundry did quite well.

[0:23:45.0] BH: Cloud Foundry did well, yeah, but I don't think that it even saw the adoption that people wanted to see that we're seen with containers. At Cloud Foundry, 100% percent has been a successful project, but not to the same extent with containers. I think it was a Cloud Foundry very opinionated PaaS, just as you mentioned, and a virtual machine marriage. I think that was a good marriage, but I think a lot of folks, they didn't want as high-level opinionation of the Cloud Foundry, but they didn't want a VM either. They don't want to like configure a VM. They wanted something more in between, and I feel like that that's really what containers — That's why containers really ended up taking off.

Then there's functions, which are closer to PaaS, but they're still not full-on PaaS. They still somewhere, to me, in between — From an opinionated perspective, somewhere in between functions as a service and containers and PaaS. That's what functions as a service are.

Anyway, it should be interesting — It'll be interesting to watch a couple things as they evolve. One; do people start to move to functions as a service? If that happens, what does that mean for a container orchestration and how quickly does it happen, and will the enterprises be able to digest it? Just as you were saying earlier, enterprises don't even know exactly what to do with functions as a service just yet.

But the reason why I think that's so interesting is because as much as you know — As I said, a lot of people didn't digest PaaS well, there's something to be said for the simplicity of the PaaS model.

[0:25:23.9] JM: And the success of Pivotal and Cloud Foundry.

[0:25:26.3] BH: And the success of Cloud Foundry. Hence, I think this interest in functions as a service, this sort of like, "Well, because the container world, it is getting more complicated, right?" I mean, doing a deployment these days on Kubernetes, sometimes the joke is just like, "How many YAML files do you end up having for doing the deployments?" and it's sort of kind of —It's sort of this victim of its own success. It started with a simply YAML and then all of a sudden you had your persistent volumes YAML and your persistent volume claims YAML and you had all these YAMLs that are being constructed, and while that's fine for the low-level hacker community and the folks that are willing to really engage and invest. For a lot of enterprises out there, that's like, "Well, hold on a second. What just happened to me being able to be in my git repo and type git push and the system building, whether it's Heroku or any of the PaaS's, getting my build pack going and pushing it up to the cloud and just getting it running. That's all I want at the end of the day, right?

So it'll be interesting to see how that evolves in the container orchestration community, which I think it is. It'll be interesting to see if PaaS comes back with a vengeance, or if there's more PaaS like things that happen in the container orchestration community on Kubernetes, or if people start adopting functions as a service, serverless, more aggressively specifically to simplify for a lot of the enterprise developers how they can take advantage of this dynamically scalable, elastic and dynamically scheduled and blah-blah-blah environment, all the things that containers are giving us, but they just don't want to go through the complexities necessarily of creating the containers. So it'll be interesting to see how that evolves.

I always loved the name serverless, because that's containers, right? I think that was the whole point, right? It's like, "We're not giving you servers or VM's. It serverless from the perspective of whether we're going to schedule the containers on this server or that server, it doesn't matter. Whether that servers is virtual or physical, it doesn't matter." I was like, "It was kind of the serverless before serverless," and then, to me, it was very telling when functions as a service first came out and everyone called it serverless, because I was like, "Wasn't that what we're doing with containers, right?" To me, that was very telling that like, "Okay. There's still people out there that feel like, "Well, that's even still too complicated. Serverless needs to be even more abstracted," and to do that we really need functions. Of course, than the joke is once we have serverless, then maybe we'll get to functionless, and I don't even know what that is, but —

[0:28:15.7] JM: It's all UIs, UIs, Windows, VR interfaces.

[0:28:20.6] BH: Who knows what's happening there?

[0:28:21.7] JM: Yeah. Maybe we'd talk about that a little bit later. I want to ask you about the container orchestration wars, because you were in the heat of battle when that was going on, and I was doing coverage of that, and I was going to conferences too. My sense during the time was especially looking back in retrospect was — At Cube-Con, seeing how many enterprises are interested in buying some Kubernetes related solution as supposed to two years ago when I got the sense that a lot of enterprises were like, "I don't know what to buy, because there's Docker, and they've got their own company, and they've got their own orchestration system, and then there's other orchestration systems, and then there's this new one called Kubernetes that seems to be gaining steam. I'm just going out hold all my bullets back if I'm the hospital, or the telco, or the bank." So the community as a whole, the orchestration community/the Docker community was like, "Ah! We need to resolve this, because there are so much money waiting to be spent," which you can see now at Cube-Con. That money is being spent, and it's great, and now the community is bonding together and it's working out its contentions and special interest groups and working groups and seems like the diplomacy is there, and it's great, and everybody is happy. But at conferences two years ago it seems like there was a little more acrimony, a little more despondency. What are your reflections on that time?

[0:29:50.5] BH: Yeah. Well, so we were definitely — Were in the heat of this. We've founded Mesosphere around Apache Mesos, which is not a container orchestrator, but it is a system upon which it's very easy to build container orchestrators. We had one at Mesosphere called Marathon, and a bunch of different organizations had their own. Netflix, in fact, just a week or two ago, maybe three, maybe more, a couple of weeks ago, I'm not sure, they just open-sourced their container orchestrator that they have on top of Mesos that they been running for years, and they run millions of containers on. There's been a bunch of other ones as well.

When Mesos was first designed, it's interesting, it’s kind of bringing this whole thing full-circle from our original conversation. When Mesos was first designed, it was designed with this expectation that in a couple of years people would want to have multiple different ways of scheduling, orchestrating containers, and that we should separate out the resource management side of things and the orchestration side of things, and maybe there'd be crazy new programming models like, say, functions as a programming model for people that would need to get orchestrated somewhere across the cluster.

So Mesos kind of made this deliberate decision. We’d made this decision of separating out the two layers, do resource management in one place and do the orchestration in another place, and that way you can go ahead and you can orchestrate everything from containers to Spark tasks, to Hadoop maps and reduces, to functions for functions as a service, whatever it was.

[0:31:31.4] JM: By the way, when you say resources, you mean —

[0:31:33.6] BH: CPU, memory, GPU's, Disk I/O, disk themselves, whatever it is. Any kind of resources that might need to be consumed for doing some kind of computation. Anyway, it comes back to that question earlier about what's going to happen? Is Kubernetes going to bring in serverless or are they going to keep it running on top? Obviously, I’m biased because of the system that I built in the past. We made that decision early on that it should be outside and run on top to keep the core simple, and I think that Kubernetes maintainers are going to have to ask those same questions.

But on the flipside, for us, what ended up being difficult as you asked me to reflect on the PaaS, what have been difficult for a lot of people was there wasn't a single way of doing container orchestration on Mesos. In fact, at the height of container orchestrators on Mesos are probably five. There was Marathon, there was Aurora, there was one called Singularity, there was Titus and there is one more that I'm forgetting. Anyway, there is fifth one. There’re five different ways of actually doing container orchestration of top of Mesos, which on the one hand is sort of like a powerful thing, but on the other hand is very difficult for a community because they're like, “Which one should I pick?” In retrospect, that was sort of — Architectural, I think it was a great thing and sort of for everybody else it wasn't a great thing. Whereas Kubernetes, was — They’re like, “Okay. We should have one way of doing container extraction. That's it.” And I think that was a strong thing for helping people to be like, “Okay. Well, there's a lease just as one way we can actually do it.”

Now, for us, and it’s the Google way, which definitely lends a hand, as we all know. They have a —

[0:33:29.6] JM: There’s some sample sizing.

[0:33:30.4] BH: They’ve got a bit — Exactly. That’s right. Well, it’s interesting when you say the Google way, because I think we should clarify. It's not what Google runs, right? The interesting part about that is in the early — When I was at Berkeley we’d actually spent some time having great discussions and debates through the Berkeley Google research relationship about what was the best way of doing scheduling container orchestration, and their initial system, which was Borg, they ended up publishing a paper later around their other system called Omega, and the Omega model is actually much more similar to the Mesos model of like, “Hey, let's actually see if we can separate these two things out.”

[0:34:13.8] JM: Oh! That’s the two-layers scheduler, or the dual schedule. Whatever they called it.

[0:34:18.4] BH: Exactly. What we first did with Mesos was exactly that. We did two-level scheduling. That was this whole thing of like the resource management is one layer, and then the schedulers, the orchestrators, if you will, for all the different kinds of apps on top were the other ones. That's then what Omega did as well. So even in Google, I think they were recognizing that they needed to rethink this sort of monolithic style that they'd had for Borg and figure out how they might want to improve that, and that was the Omega work that they actually did. To the best of my knowledge what end up happening is that Omega work ends up getting pulled into Borg. They took some of the best ideas from that and basically then made some tweaks and changes to Borg to reflect those, that work that they had done in Omega. But regardless, Kubernetes definitely has strong backing from Google and I think that goes a long way in our communities most definitely.

So kind of fast forward to today, one of the benefits for us, at Mesosphere, continuing to use Mesos for the work that we're doing is because of this two-level model. We can bring really complicated distributed systems into the system. So that's what we did, right? We embraced Kubernetes to allow you to run it on top of the platform just like you can run a bunch of these other complicated systems; Spark, Cassandra, Kafka, Elastic, yadi-yadi-yada.

So we always — Mesos has always been this lower level substrate for us, and so the container wars, I think, it was this tough thing because when everyone's like, "Oh! The container wars!" Everyone is really pitting mesas against Kubernetes, against Docker Swarm, and it wasn't exactly the right comparison, right? Mesos had the name, because when we talked about it at Twitter and when we talked about — These other big companies that are still huge users of Mesos, whenever they talked about it, they talked about it as Mesos, but really the way they're doing container orchestration was their own thing that they built on top of Mesos. That's where you ended up having these things where they are getting pitted against each other.

From our perspective we're like, "Well, this is a low-level thing," and the real container orchestration that's happening is on our system on top called Marathon. If people don't want to use marathon, they want to use Kubernetes instead. Great! That's totally fine. So we brought Kubernetes into the system so you can uses it as well. That's great because we end up — That decoupling actually lets us do some really powerful thing with Kubernetes, like run multiple instances of Kubernetes at the same time on the platform, which people are like, "Why would you want to do that?" I'm like, "Well, you might want to run a dev instance and a prod instance." "Why would you want to do that?" "Well, you might want to run the latest bleeding edge of the codebase in your dev cluster, and so you can be testing —"

[0:37:09.7] JM: [inaudible 0:37:09.6] Kubernetes codebase.

[0:37:10.8] BH: Kubernetes codebase in your dev cluster. Exactly. In prod, you might want to run back a few versions. So if you run into issues in your dev cluster and you know that you could —

[0:37:22.6] JM: Like the Manzo presentation yesterday. I don't know if you saw that.

[0:37:25.3] BH: I didn't see that one, but yeah.

[0:37:26.7] JM: There were some — They updated Kubernetes and it changed — If you have no instances of a service running, it used to be empty JSON and then it got changed to null object, or whatever the — What's the JavaScript null thing?" Then they got NPE because of it, because they had updated Kubernetes aggressively, and they hadn't changed their services so they're like, "This is not going to have a problem," but this illustrates the value of exactly what you're saying.

[0:37:54.4] BH: Exactly, right. Yeah. Again, as projects — Kubernetes has been a fast moving project. I think it'll probably slow a little bit because stuff like this will cause issues at Monzo and even bigger places as well, and that will — It will force the project to slow down and operate a little bit differently because they'll have to, because if they continue to have this kind of volatility, ultimately people will be like, "Hey! We can't reliably run this stuff."

Anyway. Yeah. It's not just that you might want to have two of these clusters. Some organizations might want to even provide something like Kubernetes as a service to their internal teams, and if they go to the public cloud, they could pretty easily do that these days. You can go to any of the public clouds, Kubernetes cluster for every single one of your team. That's really, really easy.

But if you're on-prem, that's not as easy, right? How are you going to do that? Well, you could spit up a Kubernetes cluster manually for each of these teams, use Puppet, Chef, Ansible, but then you're sort of in the day zero operations of that Kubernetes cluster, and that's it. Everything else you are really managing.

You could run open stack and then run — Bring you up to these Kubernetes clusters on open stack but then you basically — You've adopted containers to get away from virtualization and you're back in this world of running virtualization. So that ends up being one of the other things that we're excited about running Kubernetes for us on DCOS, is Kubernetes in containers. So there's no virtualization, and so in it's actually nested containers, because the Kubernetes clusters themselves are running containers. So we've had to do a bunch of really interesting workaround nested containerization, which has been fun.

[0:39:43.0] JM: Any performance penalties for that?

[0:39:45.0] BH: Not performance penalties. No. No. It is interesting though. There're only a few places left in code these days where you can have Constance, and that's the Linux kernel, right? Most of the code we need to write these days. We can't be like, "Oh! 32. Let's not have more than 32," but I'm pretty sure you can have more than 32. I can't remember the exact number, but you can have more than some power of two nested containers in them in Linus and I think it's 32.

I mean, I don't know who needs 32 nested containers, but I just — II love that, that at some point in our stacks as you get lower and lower eventually, there's just limits. Everywhere else we just try to make it like, "Oh! We'll build the software so you can have an infinite number of these things," but at some point that would be too much memory to store and in the Linux kernel, so they say, "No. 32." "Why 32? Because I can fit that in a couple of bytes and— "

[0:40:34.7] JM: Not those guys are limiting the block size on Bitcoin. This is the same Linux people.

[0:40:40.2] BH: That's right. Well, there's going to be Constance and limit someplace.

[0:40:42.5] JM: Yeah. This on-prem market, is that where you're finding the most success in terms of Mesos customer base?

[0:40:49.4] BH: Yeah. I mean, first for DCOS is the product of Mesosphere, also open source project, and on-prem is a huge, huge market for us. Most definitely. Again, I mean, we're working with some folks that are just interested in running just Kubernetes, but they want a way of running that where they can run multiple versions of it. They can easily do upgrades of it. That's, again, because the way we run Kubernetes on top, the day two operations of it is all part of the Kubernetes experience, the Kubernetes as a service experience on top of DCOS. We have software that's managing the upgrade of Kubernetes itself, which can be at a painful process. It has many stateful components. There's a lot of work that actually has to be thought through when you're actually doing that.

It’s a really interesting thing, and getting back to your earlier question, container wars. Yeah, sure, and everyone — Having now container orchestrator that we can all work with our customers on, that's a great thing. That’s wonderful to be able to bring Kubernetes via our platform to customers as well.

It'll also be interesting for us to bring this full circle as well to see what happens in the industry when it comes to — Especially in the enterprise, when it comes to the serverless stuff, because we do have more people starting to ask for it, and it will be interesting if we find ourselves bringing functions as a service to our platform, just like we brought Kubernetes as a service to our platform and sort of what the spread will ultimately be between those two.

[SPONSOR MESSAGE]

[0:42:31.1] JM: At Software Engineering Daily, we have user data coming in from so many sources; mobile apps, podcast players, our website, and it's all to provide you, our listener, with the best possible experience. To do that, we need to answer key questions, like what content our listeners enjoy. What causes listeners to log out, or unsubscribe, or to share a podcast episode with their friends if they liked it. To answer these questions we want to be able to use a variety of analytics tools such as mixed panel, Google analytics and Optimizely.

If you have ever built a software product that has gone for any length of time, eventually you have to start answering questions around un analytics and you start to realize there are a lot of analytics tools.”

Segment allows us to gather customer data from anywhere and send that data to any analytics tool. It's the ultimate in analytics middleware. Segment is the customer data infrastructure that has saved us from writing a duplicate code across all of the different platforms that we want to analyze.

Software Engineering Daily listeners can try Segment free for 90 days by entering SEDAILY into the how did you hear about us box at sign up. If you don't have much customer data to analyze, Segment also has a free developer edition. But if you're looking to fully track and utilize all the customer data across your properties to make important customer-first decisions, definitely take advantage of this 90-day free trial exclusively for Software Engineering Daily listeners. If you're using cloud apps such as MailChimp, Marketo, Intercom, Abnexus, Zendesk, you can integrate with all of these different tools and centralize your customer data in one place with Segment.

To get that free 90-day trial, sign up for Segment at segment.com and enter SEDAILY in the how did you hear about us box during sign-up. Thanks again to segment for sponsoring Software Engineering Daily and for producing a product that we needed.

[INTERVIEW CONTINUED]

[0:45:00.9] JM: To push us pretty far into the future, I had a bunch of questions about storage, but I'm going to skip those right now. This IoT stuff that's coming up, I've been trying to cover this as best I can, but you can't really cover it because it's not a reality yet, but the world where you have a drone flying around outside, and a self-driving car going down the street outside your window, and smart streetlights, smart mailboxes, smart refrigerators, all of these smart infrastructure that has a lot of idle CPU time and unused resources. This seems like a great place to schedule various tasks on to, right?

You've got some job you want to run essentially on your phone, like something that would be very complicated and today you would have to call it and it would go get run on a data center somewhere. You can imagine it getting distributed among the drone, and the self-driving car, and the refrigerator, and the stove, and all these different computers you have lying around not doing anything. I'm just wondering if you have any vision for what that's going to look like or what kinds of — Because that seems like the kind of thing that serverless is what is well-suited to, "Okay. I have this thing I want to get done. Go schedule it for me."

[0:46:20.0] BH: Find some resource.

[0:46:20.8] JM: Because that's what serverless does, is it just abstracts away all the complex scheduling work.

[0:46:27.2] BH: Yeah. I mean, I think we probably have a couple more steps in the IoT world before we get to that one, but it is an interesting question. I mean, the immediate concern — Not concern, but one of the immediate things that has to get figured out is who owns these resources, and is it — If I'm using them, these idle resources, how can I basically trust the computation that's happening on those resources, right?

If these are, say, my resources, like I've got my drone that's flying above me, I've got my self-driving car, I've got my watch, I've got my phone, I've got — These are all my resources, then I think that's one thing for me to be able to use all those resources in a really interesting way to run some distributed computation on my behalf to do something interesting, that it would be fascinating, right? I've got the drone that's not doing any interesting computation, like whatever other computing resources I have that are part of my own collection. I want to do some interesting stuff, and I can farm that out and do it.

The question I think gets a little bit harder when it's like, "Okay. Now I've got these CPUs in stoplights and streetlights and mailboxes," and what I think is interesting is, well, if the government owns it — or maybe it's not government — or maybe some private entity, depends which country I suppose you live.

[0:47:45.9] JM: It's Amazon. Let's be honest.

[0:47:48.4] BH: So that's interesting, right? Because if it's Amazon, then this is like — I don't know. Cloud 2.0. I hate to say something like that, but I could totally see Amazon, because they own the machines you're running when you're in the cloud on. They own those. You don't own those. You "own" the VM, but really — I mean, the reason why TPM and trusted computing is such a critical thing for a lot of organizations is because they want to know. They're able to trust the computation that's actually, that's happening, that there's nobody interfering and tweaking some calculation, which is actually showing that their balance sheets or their annual budgets are dramatically different than what they actually are.

That will be a problem that has to get solved when we move into that world. But what I think is interesting about it is I think the first phase will be edge computing, which is the idea that you have machines in your mailboxes and in your — By mailboxes, I'm thinking of like the blue in the states, public mailboxes that you drop stuff in, as well as in your parking meters, as well as in your streetlights. I think that will be phase one, and we're already arty starting to see that, this notion of the edge cloud, if you will, edge nodes becoming a much bigger thing. One of the organizations that we work with, Royal Caribbean, their edge data center is in a cruise ship, which the coolest thing to think about ever, right? A cruise ship, has a data center in it to do processing. Why? Because they want to be able to do interesting things on their cruise ships, and that's great because not only is it an edge data center, but it's an edge data center that moves, right? So it's much more like the self-driving car, or the drone, or anything else versus, say, like an edge data center at an airport where when an Airbus A380 lands, the double-decker big planes got terabytes worth of sensor data that it wants to do some interesting processing with, that datacenter is probably not moving. The plane is landing. You plug in the gas, you plug-in the fiber and then you do some processing in probably literally a shipping container that's been placed on the tarmac. Who knows? Maybe we'll even be putting racks in A380s at some point. So you can even do the processing on the machine and then will even more edge. It will be a moving data center, moving edge data —

But I think that will be the first phase, is we'll do a lot of this really interesting computation, more and more interesting computation on the edge. I think one of the special thing that's interesting, that is we've got a couple of pieces of software that's going to — Not software. Hardware that's going to make that even easier, things like GPU, things like FPGAs and you're going to see even more hardware that's going to make it easier for people to do really sophisticated computing at the edge. Whereas like maybe in the past they would've been like, "Well, the reason why I have to go to either my data center or my cloud is because I need this much computing resources."

You can pack a lot of GPUs into a single machine. You can pack a bunch of machines into an actual shipping container on a tarmac of an airport. That's a lot of computation you can already start to do right there, and now you don't have to ship the data. This really is a lot of data. I mean, it's a crazy amount of data. We talk about the amount of data that's generated by the

CERN collider here at Cube-Con and you're talking almost the same amount of data, terabytes worth of data that's generated.

[0:51:27.2] JM: So you can just think about once you have a sensor in your underwear and you can take samples of your blood from your shirt collar and stuff like that. That's going to happen. That's going to be a ton of data. I don't want to have to o send all that to the data center.

[0:51:42.7] BH: You don't. No. You're going to want to be able to do some processing of that much more locally so you can do some things in real time. I think that's going to happen and that's going to be the first phase. Then I think — I don't know. I don't want to call it cloud 2.0, but I then I think that once you start to get all these resources, exactly as you said, out there, there'll be this interesting thing, because I think actually we'll see a lot of people move computation at these places, and then there'll be an interesting question of, "Do we have a way of selling that extra capacity in the edge to people? Just like Amazon sold their extra capacity in their data centers. Will we have this whole — Will it be Amazon or will it be somebody else that starts gobbling up all these edge locations to be able to sell virtual machines, containers or functions as a service for people that want to do interesting computation in the edge? That to me could be — Who knows? The next phase, but the whole edge thing, I mean, that's happening already. If there's any trend that we are seeing in our customers, that's one of the biggest ones.

[0:52:46.0] JM: What customers are you talking about? Oil rigs?

[0:52:48.3] BH: .Oh, yeah. I mean, yes, exactly. So heavy machinery customers that want to send their heavy machinery to jobsites, to their own customers, where they want to increase the efficiencies, the yields, the safety, whatever it is of that stuff. They want to be able to do the processing right there. There're some folks that we've been chatting with where they're like, "Well, when we get the alert, we will get an alert, but when we get the alerts, that's like too late. If the machine is telling us like, "Hey, we've hit this alerts." It's like, "Hours ago we should've started to do some other things to try to improve this. We've either lost yields or it's a bad situation, right? I think there's definitely an opportunity for us to bring computation to the edge to do this.

In some of these places, they don't even have great connectivity back to a cloud. They can't be shipping that much data. Even self-driving cars and connected cars these days, they're on LTE and they're pushing some data up on LTE and they can send quite a bit of data, but not nearly as much data as they are actually collecting from their sensors. There's some folks that we work with who they talk about their test cars that are out there on the roads, not the ones that customers have purchased or used, but test cars. When they pull into the test shop, they pull out the hard drive and they swap in a new one and I'm like, "Oh! And then do you copy all that data from the hard drive up to your data centers?" They say, "No. The UPS at the data centers." Because it's so much data, and they want to do all the processing. They want to pull all these stuff in.

[0:54:24.1] JM: It's Netflix.

[0:54:25.1] BH: What's that?

[0:54:25.8] JM: It's like Netflix.

[0:54:26.4] BH: It's like Netflix there.

[0:54:28.4] JM: They're sending the DVDs.

[0:54:29.6] BH: They're sending the DVDs. This stuff is way cheaper for them to UPS or FedEx, or whatever it is, the data. So if wherever the car shows up, they could actually do the processing right then and there. That's really powerful. They could do the processing faster, and especially in this whole world of AI and ML, some of the most important parts of the processing is training the models with that data, and then if you don't need as much of the data, you've trained the model. You can keep training the model overtime, but then maybe it's easier to share the models and not have to share all the data. So if you could do that processing, if could do the training right there, update your model, send it someplace else.

One of the things that I think actually will be a really interesting question, if we ever get to a world where we are doing more computation in the edge, on devices, especially when the computation and data, there’s an ambiguity about who it's owned by that. I think that will be an interesting world, right?

If I've got my car and my car is collecting data about my driving, and then the car manufacturer is doing — They’re doing processing on that data on, because they want to provide some either better service back to me or somebody else. We’re going to have the same sort of problem of like who owns the data and who owns the outcomes of that data. Is it me because I was driving the car? It's the same thing about like when I go surf on Google, or surf any of these other places and data gets generated about my actions and what I'm doing, like who owns that data. I think we’re going to have that in spades even more as we start doing even more computation in the edge and collecting even more data about what we’re doing in the edge.

I foresee even more Supreme Court cases in the future about who owns the data. Is it me? And people that own these edge processing devices, can they use that data to — I mean, here's the crazy one, right? Can they use that data that it collected about me to generate a model, a machine learning model? If they do, could they then sell that model to some other person because it's a great model, because it's been based on all their customers data? Do I actually own that model because it came from my data, or does — Anyway, interesting questions that all will have to be answered as we moved to this future world. But I think we’ll answer them and it will be really fun to see what we can do.

[0:56:51.7] JM: Yeah. I think so too. Maybe we could do another show, and that's closer to reality. So I know we’re running up against time, but I wanted to talk a little bit about storage just because I saw you at the storage working group yesterday. So the question of storage, if I understand it correctly, is people think of containers as a place to run their stateless applications for the most part. That's not to say that there is no state on a Kubernetes cluster. You obviously have state. You have mostly in etcd as far as I know. That's like the state of the Kubernetes clusters maintained in etcd, at least. But as far as having statefull, long-running applications, I think people are not using Kubernetes, for example, to host their own databases as much as they are using Kubernetes for applications that are interfacing with a database that's in Amazon, like RDS or interfacing with S3 or something, and there is a lot of desire to get database and database-like solutions, object storage, etc., file storage, block storage on a Kubernetes cluster so that building blocks can be built in a Kubernetes cluster agnostic of a certain cloud provider. Am I understanding the motivation for working for the storage?

[0:58:13.9] BH: Yes. I think a lot of people talk about containers, container orchestration and Kubernetes as being the thing that can start to give people cloud portability, but how can you really get cloud portability if you're still tied in and locked into the storage APIs on whichever respective cloud you're using? If you're using S3 and you're putting your data there, that's where you're at, right?

I mean, as much as you might be running your containers by Kubernetes, you're not moving to another cloud unless you're willing to deal with the latency overhead of talking to S3 from the other cloud, or you're willing to copy the data or move the data. It's an interesting thing too, because there's the API lock-in which is not as much of an issue with S3, because S3 has become much more of a de facto API.

With some of the other storage services, the as a service products that you see on the cloud, there definitely has the API lock-in issue, but there's even more of just like a data gravity and a data mass problem, which is like the data is there. You either got to copy it or you got to move it.

So for the first one, for sort of the API problem, I think being able to just pick your own API and your own service that you're going run on a container orchestrator or some system that gives you the flexibility of being able to move between clouds is a huge desire, just exactly as you said. Again, you still have the data gravity thing that you got to think about. So it'd be interesting to see how that evolves.

Now, I love the clouds and everything that they've done, but what should be interesting is to see how they really feel about the fact that people would want to get portability from these services, because it's definitely not in their best interest from a business model perspective to want to help that in the open source community.

The reason why I'm saying that is I think one of the reasons why storage has also not been as much of a focus in a lot of the containing orchestration environments is because whose really incentivize are aligned to make a bunch of that that actually work? Even the big storage companies, a lot of them would prefer that you kept using their storage solutions, then use some piece of software that you might run and manage yourself in a converged way, because maybe over time that could give you the flexibility of even not having to rely on their services. Just interesting questions to be asking and thinking about when it comes to storage.

Now the reason why storage — That's all kind of the business and industry reasons. The biggest reason why storage-based systems haven't been brought to these platforms from a technical perspective, not from sort of an incentive perspective or anything else, it’s just because it's hard, and it's hard because with a stateless container, when it dies, I can reschedule someplace else. It's not a big deal.

Most likely with a stateless container, I'm running many instances of it. So if anyone single one dies, I could reroute traffic to somebody else. There are a lot more flexibility that you actually have. With statefull, I probably have to solve the; am I using local storage or am I using some external storage problem first?

If I’m using some external storage, it gets a little better because at least I don’t have to deal with the storage on my — But it's still hard, because the ways in which you might want to do upgrades or the ways in which you might want to do things like backups or restores or like anything else that's kind of complicated day two operations as we like to think of them for a lot of these services. There’s just way more complicated than these stateless micro services. They don't have day two operations for the micro services. They just have like reschedule, bring in a new version of it, whatever it is.

So there is that whole aspect, which is the complexity of just operating it, but then there's the whole — You can use local storage versus external. If you choose to use local, basically what you're doing is you're taking a system which was designed to — You’ve heard the phrase; cattle, not pets, right? Which was designed to treat all the machines as cattle and you're effectively creating, making some of the machines pets. That's kind of like it's counter to everything we've been trying to do if you’re going to be using local storage, because you're making those machines pets, because that's the machine where you wrote the local storage, which means that's the machine where you need to reschedule the stuff in the future, which means that's the machine where you can't reschedule anything else in the future. Let's say you're doing an upgrade and you bring down the container that's trying to do storage on that machine, you can't let somebody else jump in and start running on that machine, because you have to be able to rerun this thing on the machine. There’s a whole host of other problems that basically end up happening with statefull, which ends up being complicated.

Again, because I think anyone who is running Kubernetes in the cloud, they could solve this problem using external storage, like S3, even external volumes like EBS. Lots of ways in which they could actually go ahead and solve these problems, I think that there hasn't been as much of an incentive to actually try and figure that out, coupled with the fact that there’s not great incentive. You have a really hard problem. It’s going to be, I think, longer until any of that stuff actually evolves.

Now, it's interesting, because one of the things we did — If I could give a small plug here for second for the work that we’re doing in DCOS, that was one of the first things we actually tackled with the things on DCOS. Our Cassandras and our Kafkas and our HDFSs and Elasticsearch and all these things, which are heavily statefulf systems. We said, “Well, we want to we want to figure out how you can bring the container-like experience or the public cloud-like experience for those things onto the platform and make that as good as just launching containers.

That’s one of the things that we did really early, and that's again why from our world we have — You can bring up and run Kubernetes alongside of HDFS and Cassandra and Kafka and Redis and all these things. So we just — In Mesos itself, we spent a lot of time investing and what are the parameters we need to do to be able to have things like reservations for these stateful services and to be able to think about what's the best way to manage the discs and so forth and so on, which is one of the things that ultimately inspired us working with Google and other people on CSI, the container storage interface, which is where you saw me yesterday at the storage working group, and we briefly talked about that at the storage working group. There's other talks here on CSI at Cube-Con as well.

Yes. That focus, that was us thinking about how’d we want to do that and then figuring out if there's things we want to contribute back to the open source community when it comes to storage and what we’re trying to do.

So you will continue to see this push in the Kubernetes community for people wanting to think about and do storage on Kubernetes, but there's a lot of things that I think will have to happen over the course of the next couple of years for that to become a reality just because it's a hard problem. Just because there's a lot of work that has to get done, and what will be interesting is to see who from the industry really steps up and helps to drive this. Again, this just me coming full circle with what I was saying earlier in this part of the discussion. Will you see it happen a lot from the cloud providers or not? Because if you do, I mean, hats off to them, but in many ways that's potentially cannibalizing their business, because for you to run MySQL on Kubernetes instead of using the clouds relational database, definitely gives you more flexibility of being able to walk away from that cloud.

[1:05:59.8] JM: I mean, it's going to be Google. If anybody is going to push it, it's going to be Google. In terms of having the vector with the most magnitude would be my guess, because they've got the most market share to recover.

[1:06:14.5] BH: Yeah, and that'll be interesting because –

[1:06:16.9] JM: I don't even think the other cloud providers are going like really – They won't be resistant. I don't think they'll be resistant to it. In fact, I think they'll even provide some anti-frictional, minor anti-frictional capacity just because – I mean, I think there's a lot of ways to make money. There's a lot of ways for cloud provider to be successful, and you don't want to build a bad reputation. The cost of a bad reputation is just getting higher and higher and higher from the cloud provider's standpoint. I think it's just like we'll see more good actors or more increasingly subtle bad actor behavior if it's going to be bad actors.

Anyway. Ben, great talking to you. This has been great.

[1:06:53.6] BH: Yeah, thanks for having me.

[1:06:54.7] JM: Yeah. We should do this again sometime.

[1:06:55.9] BH: Sounds good.

[1:06:56.9] JM: Okay.

[1:06:57.1] BH: Cool.

[1:06:57.4] JM: Awesome.

[1:06:57.9] BH: Okay.

[END OF INTERVIEW]

[1:07:01.3] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes. You can quickly provision clusters to be up and running in no time while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked into any one vendor or resource. You can continue to work with the tools that you already know, such as Helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications off-line. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes. That e-book is available at aka.ms/sedaily.

[END]