EPISODE 671

[INTRODUCTION]

[0:00:00.3] JM: A company runs a variety of distributed systems applications, such as Hadoop for batch processing jobs, Spark for data science and Kubernetes for container management. These distributed systems tools can run on-premises, in a cloud provider, or in a hybrid system that uses on-prem and cloud infrastructure. Some enterprises use VMs, some use bare metal, some use both. There's a lot of complexity in all of the different configurations that a company can have.

Mesosphere is a company that was started to abstract the complexity of resource management away from the application developer. Instead of a developer managing virtual machines, provisioning cloud infrastructure, or wiring all that infrastructure together to run a distributed application, the developer spins up distributed applications, like Kubernetes, Spark, or Jenkins on top of Mesosphere. Mesosphere standardizes the provisioning and the setup of the machines of the underlying infrastructure.

Using Kubernetes on top of Mesos, allows you to separate the resource provisioning from the actual container orchestration. In a previous episode with Netflix, we explored how Mesos can be used with a container orchestrator on top of it to simplify that resource management for things, like microservice deployments, as well as data science workloads, basically anything that you need a container for.

Chris Gaun is a product manager at Mesosphere, who helped build Kubernetes as a service from Mesosphere. In today's show, he explains why it's useful to have separate layers for resource provisioning and container orchestration. He also talks about the difficulties of manually installing Kubernetes and why Mesosphere built a Kubernetes as a service product.

Full disclosure, Mesosphere is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:02:17.6] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries, speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly.

Transifex is a SaaS-based localization and translation platform that easily integrates with your agile development process. Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowdsource your translations.

If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial. With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time.

Translators work on live content within the development cycle, eliminating the need for freezes, or batched translations. Whether you are translating a website, a game, a mobile app, or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15-day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW]

[0:04:05.9] JM: Chris Gaun is a product manager at the Mesosphere. Chris, welcome to Software Engineering Daily.

[0:04:09.8] CG: Thanks for having me.

[0:04:11.1] JM: I'm looking forward to talking to you about Kubernetes and Mesos and a variety of distributed systems applications. I want to start with the discussion of a typical modern enterprise. If you talk about a big modern enterprise, like a bank, or an insurance company, or Twitter, or Yelp, they have a variety of systems that they're dealing with. They have some software that's written from scratch, they have some cloud provider technologies they're using, they have some open source tools that they want to run, they have Spark and Hadoop. Describe how a typical enterprise is managing this variable different types of infrastructure.

[0:04:49.5] CG: Yeah. When we're talking about a typical enterprise, I would say you would cast them as probably the Fortune 500, or Global 2000. These are very large organizations. Number one, they typically have a strong central IT. Meaning that each line of business, which will have developers will go to central IT for their infrastructure. Now that used to be delivered in-house with they put servers on the rack. More and more that's going to the public cloud and more and more, that's going to multiple public clouds and a mix of both on-prem and multiple public clouds.

That's where they're at right now. Now as far as the applications that they're running, people might not know this, but large banks have thousands of enterprise developers and often, thousands of applications. Most of their applications, most of the applications that they wrote themselves are legacy applications up the 80%, 90%. It's about 70% of them are in Java.

Those applications are running on a variety of different technology stacks, typically old middleware and sometimes on something that's even more legacy like a Unix or mainframe. There's a variety of different stacks within an organization, oftentimes completely separated from each other, like they'll have a VDI stack, they'll have a stack for running their Windows applications and then they'll have one for running their job applications.

Now in that context within the open source world, or Java world, they're also working on several open source projects. They've used traditional databases, they've used traditional analytics tools. More and more are moving towards new fast data tools, like Spark and streaming services like Kafka, because they could get more performance out of those and they would their traditional data warehouse and reporting tools, where they do some batch workload, they get a report maybe every day that they could read and say, "This is the portfolio that I have, as far as my stock portfolio goes, right?"

They want to be able to do that on the minute, or even on a real-time basis. They're starting to build out these fast data systems. Alongside of that, there is obviously a tremendous interest in consolidating, or running applications in a more agile way. One of the ways that organizations are achieving this is through container and container orchestration, particular with Kubernetes on the container orchestration front.

[0:07:33.3] JM: What are the challenges that these kinds of organizations, like if you think about the prototypical organization, I think the bank is an interesting example, because banks typically did not start with the idea that they were going to become a technology company in the long term, but they found themselves in a situation where they are technology companies. How does the process of moving some of their infrastructure to the cloud and starting to deploy things like Spark more, more readily, and Kubernetes more readily? What kinds of challenges are they encountering?

[0:08:07.7] CG: One of the challenges that they encounter is they might be just moving stuff in the cloud in a non-discriminate way. They're not watching closely to making sure that they're getting the most efficient usage out of their cloud. When I talk to salespeople at the big cloud providers, they all say the same thing like, you could turn your server, or your cloud instance off at night when you're not using it, or during the weekend, but then never happens. Cloud usage always is a monotonically increasing function for most workloads, right?

What's happening is if they do a lift and shift, they could do that, but then oftentimes, they're running incredibly inefficiently. Their cloud instances don't have a tremendous utilization. That's one thing that you encounter and it depends on how fast you're moving to the cloud, how fast they're moving to multiple clouds.

The other thing is that they have to make a choice, like do they want to use the services within the cloud vendor and then that they do that, the API's might not be the same. For instance if they need a streaming service the streaming service from Amazon is different from Azure, which is different from Google. Some, or a lot of organizations want to be able to abstract that and actually have the same streaming service across all their different cloud providers and on-prem. That's another big challenges that they're focusing on.

Then if they're going to do that, or if they're going to do any of this, or any of these open source, fast data workloads, or even container orchestration, how do you actually manage all these fairly sophisticated distributed systems. Kubernetes is hard by itself to manage on a day-to-day basis. When you add Spark and all these other analytics tools to it, you could see that this becomes a major obstacle for adopting newer technologies for a big organization.

[0:10:06.6] JM: That first point you mentioned with the just unleashing cloud resources, or unleashing your applications onto the cloud and letting the costs ramp up. I've been going to conferences for a couple years now since doing this show. I've been going to conferences pretty regularly. I always walk around the expo hall, because I like getting a sense for how the different industries are developing. I swear, there are more and more of these cloud cost management companies that I meet.

It's an interesting industry that has developed the idea that okay, companies move to the cloud, or they start in the cloud, then all of a sudden they find themselves with all these costs and they're like, "Oh, God. How do we control this?"

[0:10:49.0] CG: Yeah. It might be the right decision at the time, because you got to remember we underestimate how much moving from one technology to the next is arbitrage, right? We always say with the better technology, I'll make you more agile, or whatever. It has all these new features. Sometimes it comes down to cost. A lot of the technologies that they're using on-prem might just be something that they bought 30 years ago, like a very expensive database.

Moving to the cloud has huge arbitrage advantage, where they could just save money in that way. Then if they're doing that and they're just moving stuff, lifting and shifting or whatever they're doing, they're not using the cloud resources in a very efficient way. Did you save money in the short run, but then they have this another new obstacle that they have to overcome going forward. That's the first thing.

Then I would say the second thing is that when they're doing this and when they're moving and when they're thinking about the arbitrage, they might be thinking about the arbitrage of what their current database cost, versus what their new database costs somewhere else. They might not be thinking about, "Oh, this is actually maybe more expensive than an open source option, or something that actually has more features out there, or something I have more control over, or something that I could use on every cloud provider." That's something that they might have to wrangle with in the future as well.

I feel that people were rational, right? They're making the right decision in aggregate right now, but then when they make this move, or when they make the move to multi cloud, or hybrid, you're going to have the next phase where they're going to find more arbitrage, or there's going to be some other ways, some newer technology, or some way that they could see based on what they're doing at that moment in time.

[0:12:40.1] JM: When these enterprises are considering some tool like Spark, or Kubernetes, they could go to a large cloud provider. The large cloud providers have managed services that can be used in instead of open-source Spark, or the open-source Kubernetes. You can use managed services. What are the pros and cons of using the managed service version, versus the open source version in some of these considerations?

[0:13:11.0] CG: I would say that you would still want the open-source version managed in some way, or you don't want to be doing Kubernetes the hard way, right? That's not something that you want to do on a daily basis. I'd also split it up and into two different things. You have those big data, or fast data workloads like Spark. For those, the thing to think about is if you do this in a cloud provider, or one cloud provider, you're going to be using their APIs for their particular service and then they're going to be using another API for another cloud provider.

It's going to be different on how you actually use Spark, or their version of Spark, or their version of Kafka from cloud provider to cloud provider, right? Enterprises are going to think about that, right? On Kubernetes front, it is a little bit different, because your applications should work on any Kubernetes cluster as long as it's a Vanilla Kubernetes and the underlining infrastructure offers the same features, like for instance if you are using, I don't know, like GPUs in Kubernetes and if you want to switch to another provider, or another solution, then you have to find another solution that also offers GPUs.

In Kubernetes, it's a little bit different, because all the API should be the same. Now where it is – where you fall into the same track as you do with say the managed services for the big data and fast data tools is that everything underneath will be different. The way that you manage Kubernetes will be different from each provider, from each solution, from each software. right?

You could say that my app should work for moving from one Kubernetes cluster to the other, but how I actually manage Kubernetes is going to be completely different across providers and software. I do think that organizations should think about that and should think about okay, going forward, if I actually did want to use multiple cloud providers, or use this on-prem and another cloud provider, how can I get a consistent way of doing that, both for the application front, so for the APIs and underneath southbound on the management front of actually how I'm managing Spark, or Kubernetes?

[0:15:27.9] JM: I first interviewed Ben Hindman, who's the – he's the creator of Mesos and one of the founders of Mesosphere. I interviewed him four years ago, and this was when I was really – I didn't understand the space at all, I didn't understand – well, nobody understood the space super well, but Ben was certainly on the leading edge of it. It was interesting, because then I interviewed him again, I guess six months ago or so. It's funny, because his vision was pretty consistent. He was very prescient about the idea that – I think his idea of Mesos was here's a distributed system for running your distributed systems. This was born out of his experience in the Berkeley Amp Lab when he saw Matei building Spark, or he was working with Matei and Matei was building Spark.

He was thinking, there's going to be a variety of distributed systems that people are going to want to run, and wouldn't it be great to have a simplified system for allowing you to stand up those distributed systems more easily, because it's very hard to stand up your Spark, or your Kubernetes from scratch. His idea was frameworks. Mesos and DCOS, which is the tool that Mesosphere offers, this has the idea of frameworks. What is a framework?

[0:16:51.5] CG: Yeah. Within Mesos and DCOS on the product side, there is this idea of two-level scheduling. Some of our customers have called Mesos like a meta-scheduler. It is taking care of the resources underneath these schedulers, or other schedulers. Then these other schedulers, or other frameworks have their own logic towards them – to them. It could be Spark, which has its own way of scheduling jobs and workloads, versus Kubernetes, which has a whole bunch of logic on how to actually schedule pods within itself. That's fundamentally how it works.

It's very misunderstood within the community. I should emphasize that, right? Because people will say, well, you have Mesos and that competes against Kubernetes, or that competes against something else, and it really doesn't. It's really a different layer, and it's used very differently within any technology that you would use Mesos.

People don't know this, but there's actually been – it's not widely known, but there's actually been many different container orchestrators that have worked on Mesos, so different frameworks that, if you will, that have worked on Mesos over its history. One is marathon, which was the original scheduler for DCOS, or container orchestrator, I should say, for DCOS.

Another one currently is Kubernetes, which on DCOS. Other ones that you see that are in the open source community, one is from Netflix. They quite recently wrote a blog about their container orchestrator and open sourced it, which is called Titus. Within that blog, they say they launch 200,000 instances of Titus a day. That might be a typo in the blog, to be honest, I don't know. I don't know why you would need 200,000 schedulers per day, or container orchestrators per day.

[0:18:45.9] JM: It’s Netflix, man. Don’t underestimate it.

[0:18:49.9] CG: Netflix scale, right? Well, the other thing that they said in the blog is that there’s 500,000 containers per day, but they spin up and probably spin down, right, very quickly. I don't know if that ratio makes sense. Then there's other ones. There’s Aurora, which was originally used at Twitter and is used by a few other organizations. There are other ones that you could probably Google search and find. I'm not sure if I'm allowed to talk about them, but other companies have used Mesos and talked about the container orchestrators that they’ve built on top of it.

[0:19:23.9] JM: Yeah. Well, the Netflix example is really interesting. We did a show with Netflix about container scheduling, oh gosh, I think was a year and a half ago. Tthey were talking about, was it fenso? There's something else called fenso that they used. I mean, Netflix has so much going on their dip, between data science and serving traffic and whatnot, but I think the show I did was around scheduling of data science jobs.

They talked about the two-layer scheduling, the value of having the two-layer scheduler and using Mesos together with a container orchestration framework. Talk a little bit more about that difference that they're – that you want, or under some circumstances you may want these two different systems, a resource scheduler and a container management tool. What are those two things?

[0:20:12.0] CG: You can go back to the origin of Kubernetes. It was designed as whatever you want to call it; container orchestrator, container management. It was designed, or not designed, but one of the main use cases in the beginning was to be used by a cloud provider, which Google, which had their own infrastructure as a service and they had their own internal tooling to set up Kubernetes. They have Borg underneath there IS Google Cloud.

They have a bunch of tools in order to manage the container orchestrator, in that case being Kubernetes that they're giving it as a service to their customers. I do feel that this is important, because I do think that you need, or you don't need it, but it is – I would highly suggest having a cloud provider underneath Kubernetes for similar reasons. It makes it very easy to provision Kubernetes, to manage Kubernetes on the day-to-day basis. Within DCOS which is powered by Mesos, we actually – the Kubernetes team uses it as a cloud provider.

We get resources from Mesos, like memory CPU for the Kubernetes infrastructure. The way that Kubernetes is treated in the system is it's treated like pods are in within Kubernetes itself. You got to think about that. If a pod dies, then respawn somewhere else on the infrastructure; same thing, if a Kubernetes node dies, it respawns somewhere else on the Mesos infrastructure.

If you want to draw a direct parallel, it is orchestrating the actual infrastructure of these schedulers, whether it's Kubernetes, or Spark, or anything else out there that runs on Mesos, in the same way that Kubernetes orchestrates applications, or across its pods.

[SPONSOR MESSAGE]

[0:22:15.2] JM: Leap.ai matches you with high-quality career opportunities. You are more than just your skills and a job description and a resume; these things can't fully capture who you are.

Leap.ai looks beyond these details to attempt to match you with just the right opportunities. You can see it for yourself at leap.ai/sedaily.

Searching for a job is frustrating and Leap tries to reduce the job search from an endless amount of hours, days, weeks, to as little as 30 seconds trying to get you matched to a job instantly, by signing up based on your interests, your skills and your passions. Leap works with top companies in the Bay Area, Unicorns and Baby Unicorns; just to name a few; Zoom, Uber, pony.ai, Cloudera, Malwarebytes and Evernote.

With Leap, you are guaranteed high response and interview rates. You can save time by getting direct referrals and guaranteed interviews through Leap.ai. Get matched to jobs based on your interests, your skills and your passions instantly when you sign up. Try it today by going to leap.ai/sedaily. You would also support Software Engineering Daily while looking for a job. Go to leap.ai/sedaily. Thank you to Leap.ai.

[INTERVIEW CONTINUED]

[0:23:54.1] JM: I'd to better understand the layers there. You've got a cloud provider layer and then you've got Mesos, which is getting resources from the cloud provider. Then you have frameworks that are sitting on Mesos, so you could have Kubernetes for running your container workloads for various purposes, you could have Spark doing data science jobs, you could have Hadoop doing other stuff, you could have Elasticsearch doing other stuff. Is that the different layers of the stack that you're describing?

[0:24:24.7] CG: I mean, it could be. We consider Mesosphere, the product Mesos DCOS powered by Mesos is a cloud provider in and of itself, because right, you could put it on bare metal servers. You don't necessarily need an IS provider for that. That you need CPU and memory from whether it's bare metal servers, or from a cloud provider. That needs to be offered up to the scheduler, right? Kubernetes in this case. That middle layer is what's offering it up.

You could do this public cloud providers do this in a different way this is how we do it at Mesosphere is that we use DCOS and Mesos in order to offer up those resources to Kubernetes. That's fundamentally how it works.

[0:25:13.4] JM: Somebody told me recently that going back to the bank example, banks cannot use most cloud provider state. They actually do have to do bare metal, because of regulations, I guess. Do you know if that's the case, or are there industries where you still can't use cloud provider technology? You have to be bare metal, you have to run –

[0:25:32.7] CG: Again, let's make a distinction between public cloud providers and then a cloud provider in the sense of a Mesos. In a public cloud provider, there are industries where they're more shy for public cloud providers. What it typically comes down to is for instance if you're in China, the top cloud provider might be Ali Cloud, right? Then if you're in Europe, you typically have data compliance laws, where you have to have your data in the country. You can't even leave the actual – the state.

In those cases, unless the cloud provider has a presence within the country, they might not be able to use it. Then in the case of if they want to use – so if they want to have one, say presence in the United States and then something in maybe a few European countries, and then one in China, you're already talking about the multi-cloud environment, because you would be running Ali Cloud in China and then you'd have to find out a solution for Europe to deal with their data privacy laws, and then another in the United States.

It's not that they can't move workloads to a public cloud provider. It is more, there is patchwork of compliance, there's patchwork of different laws out there and there's just a patchwork of what's available within each region, just because maybe a public cloud provider does not have a presence, or maybe they don't even – people don't realize this, maybe they don't even offer the service that you want to use in the country that you reside. Maybe you're in Europe and they don't have a container service right now. That's actually the state currently for some of the public cloud providers. That will change over time, but that's typically the reason that you see.

[0:27:32.2] JM: To some degree, it seems like Mesos can be a proxy for that changing cloud provider topology, or cloud provider versus multi-cloud, versus completely running across my infrastructure because of regulatory reasons, or otherwise. Is that a reasonable way of looking at Mesos in a modern context as a proxy for one of the giant cloud providers?

[0:27:57.4] CG: Yeah, that's correct. That's one of the reasons, and including public case studies on customers have used it. An interesting public one is a Deutsche Telekom, where they actually built a machine learning to look at their networks. They have hotspots and then they have public cellphone reception. Depending on the time of day, or where you are, it might be faster and cheaper to connect to a hotspot, but depending on if it's a busy intersection, or busy location, maybe rush hour at a train station, connecting to the hotspot might be completely – it might just stop your connection altogether, because it's just way too slow.

They built this artificial intelligence, or not artificial intelligence, I should say deep learning machine learning to do that analysis. One of the reasons that they chose to do this on Mesos is exactly that is because, it's actually all in a public cloud, but they want to move this system not only for Germany, but they want to do this in several different countries and potentially roll it out globally.

They did this on Mesos, that they could build it once and have consistency to spark Kafka and the container orchestrator throughout all the countries that they're going to when they roll it out. Maybe they find one day that one cloud provider is less expensive, or that they want to roll it out to some region where a cloud provider doesn't exist, or they don't have a particular service. That was a major reason that they went with Mesos.

[0:29:29.3] JM: Okay. That lift and shift ability, if they would have instead used just Kubernetes and they decided that Kubernetes is going to be how we're going to be doing this okay, we want to build something that's going to be in different geos, and in those different geos, maybe we don't have access to certain cloud providers so we don't want to be tied to a specific cloud provider, so we want some layer of portability. How would using Kubernetes as that layer of portability contrast with using Mesos as the layer of portability?

[0:30:00.0] CG: Okay, so again, we're talking about the two-level scheduling. Kubernetes would be an excellent, excellent way to get this portability for the different services running on Kubernetes, right, because the APIs would be the same from wherever you're setting up Kubernetes.

Now southbound on actually how you manage Kubernetes will be potentially completely different on whatever cloud provider you're choosing, or if doing this on bare metal, or if you're doing this on virtualization, and the different technology stacks, storage, networking might be not even recognizable from one provider to the next.

You got to think of the two different layers; what Mesos provides in that instance is a consistent way to manage Kubernetes across and the resources across wherever you want to run it. Kubernetes itself will allow consistency for the applications that you're running across all the different environments. Does that make sense?

[0:31:01.9] JM: Yeah. I think so. If they were to have instead used Kubernetes, they would have had to have Spark running on Kubernetes, they would have had to have any other infrastructure that they wanted to be portable to be running on Kubernetes, in order to get the same degree of portability.

[0:31:21.2] CG: Well, so adds another level of complexity. I was just thinking of the typical applications that you run on Kubernetes today. If you start running big data services on Kubernetes, you have complex systems on complex systems. You should really, before you move in that direction be an expert in both those things, right? Both the big data service and in Kubernetes itself and whatever you're going to run Kubernetes on, so three different things. That it's another level of complexity. I was just thinking of running pure whatever, a web application on Kubernetes.

[0:32:01.6] JM: Right, right, right. Okay. Then I think the contrast there is if we're comparing, running the data science workloads in those two use cases, it would be running the data science workloads on Mesos, versus them running on Kubernetes. I guess, Mesos is a more tuned for that, or how would you contrast the setup process of somebody trying to get Spark, or Hadoop, or some other what Mesos would refer to as a framework going on Kubernetes, versus going on Mesos?

[0:32:38.1] CG: Well yeah, they grew up together, right? Like Spark and Mesos. As you mentioned when Ben was in graduate school, one of the first workloads that they ran on Mesos was Spark. It's not the only big data, or whatever fast data, or streaming service that has been running on Mesos for many years. It has that maturity, it has that proven production-ready, I guess certification that you would get from just years of experience and across many different customers, or many different users.

That's the real difference. Then Kubernetes, it's a newer type of workload that you would run on Kubernetes, and it's more of a DIY experience to run on Kubernetes. I don't think anybody would disagree with that at this point.

[0:33:30.0] JM: Yeah. The difference for the Mesosphere product, DCOS versus Mesos, that's one thing I'm curious about. How do people use DCOS and how does that contrast with the Mesos I take it off the shelf and I decide I'm going to use it and just deploy it and figure it out myself?

[0:33:49.0] CG: Yes. Taking it off the shelf and deploying it by yourself, just like every distributed system, it is a decent amount of work, right? You need to be an expert within the technology, correct? Just like Kubernetes, or any of these other distributed systems. Organizations have done that, and but we're talking about the top tier of Silicon Valley, or the top tier of technology providers, or vendors that you see out there, like a Netflix for instance, or Twitter, right?

They have tons of engineers. They can work on this. They build their own stuff. This is all they do. They could get it running and tune the way they want it. Now DCOS is something that comes out of Mesosphere, which is powered by Mesos. It's the fundamental basis of DCOS. It's the glue for every framework that you would run on DCOS, whether it's Kubernetes, or Spark. It all drives from Mesos.

In this case, we made it extremely user-friendly, right? You could download DCOS, easy to get up and running, easy to get Kubernetes up and running on DCOS, easy to upgrade Kubernetes. That's the real difference. Mesos is a open source solution, DCOs is a product, right? It is productized, it is extremely well-suited to just put on a few servers and get up and running and go.

[0:35:23.2] JM: Earlier, we alluded to the issues that an enterprise might be encountering today if they're trying to get on to Kubernetes, or trying to modernize their infrastructure, even just trying to get on to cloud. I'd to understand better what you want to provide to an operator. If I'm a Kubernetes operator, or I'm a CIO perhaps and I'm picking between different technologies, I'm about to decide what the, maybe you would call a platform as a service, maybe not, I don't know, the management layer, the dashboard of the engineer who's going to be spinning up new infrastructure, or spinning up new data science. What are the requirements, because that's the product that you're – it's not exactly the product you're responsible for, because you're I think more focused on the Kubernetes as a service side of things, but tell me about the requirements for that infrastructure management layer that you're trying to offer to Kubernetes operators?

[0:36:22.2] CG: Yes. No enterprise, no user should accept anything less than a cloud provider experience at this point. It is the gold standard within our industry right now, right? DCOS is built as a cloud provider, right? It is Mesosphere's cloud provider and is the cloud provider that all our customers use. It's just not a public cloud provider, it's not something that's attached and necessarily managed end-to-end for you. When you set it up, it allows you as an organization an internal IT, we're back to the bank where they have a strong internal IT organization and they need to offer services to lines of business, this allows you to be that cloud provider.

You could use other you public cloud providers underneath as the infrastructure, you could use bare metal as the bare metal servers that you have in your data center as the infrastructure, but it gives you that cloud provider level of abstraction from the infrastructure, but still allows you to service your user, which is the line of business or developers in the same way as a public cloud provider. Now for instance when you're doing Kubernetes, you could get a Kubernetes cluster with a single command line. Okay, now you have your Kubernetes cluster, now another line of business wants Kubernetes, because they think it's really cool. Now you can spin up another Kubernetes cluster and it's that simple. They could get in in five minutes, but that's just provisioning Kubernetes cluster. That's fairly easy to do.

There's tons of tools out there that could provision a Kubernetes cluster, maybe with the security defaults out there that listeners keep in mind, make sure that it's locked down, there's 20,000 dashboards open to the internet right now according to a study that I've seen a new stack, so make sure that that's not you.

For the day too, you want to be able to upgrade Kubernetes, right? You want to be able to go from one version to another. One group might not want to upgrade at the same time as the other group. That's extremely important. If you look at a large organization, the applications that they're running might be on servers that are decades were created, or bought decades apart from each other. The reason for that is the application was designed and coded and deployed at different times, and then one group maybe never wanted to actually move from one move infrastructure.

That's incredibly important. You need to be able to offer different Kubernetes versions, you need be able to scale Kubernetes, everything that you would get if you were using a public cloud provider, right? You should accept no less.

[0:39:03.7] JM: Now does the central IT at the bank for example, did they want a central Mesos cluster, or central X cluster that they can look at and they can see who throughout the organization is spinning up infrastructure under different circumstances?

Because one thing is I've talked to some people who are at large organizations and I asked them, “Okay, so you set up microservices for your part of the bank and you have your own cluster, and then somebody else on the other side of the organization set up their own Kubernetes cluster to do something else.” That seems fine. I mean, it's like we've got different applications, just like I have a Ruby on Rails application in one side of the org, and I don't know about a Ruby on Rails application in other side of the org.

I can imagine a time when it would be useful to stitch these Kubernetes clusters together, or I don't know, maybe there's some kinds of economies of scale of combining these different clusters. Is there a desire from these larger organizations to have a management layer that can see the different Kubernetes clusters at once?

[0:40:09.3] CG: Oh, yeah. Absolutely. It's the number one thing that they want. They want a centralized tool in order to manage Kubernetes and other workloads that they’re using. They traditionally went to public cloud providers, because it offers them a way to do that. They could increasingly just do that with the software cloud provider, like a DCOS.

The reason that they want that – let's be clear, because when you're first using open source, a lot of different lines of business, a lot of different groups within the organization would go off and do their own thing. They'll start to get familiar with Kubernetes, or whatever it is and maybe get an app on it, developer in a developer stage, or development stage, and eventually what they'll do is they'll come to central IT and they'll say to them, "We now have this requirement that you need to be able to run Kubernetes, so that we could move our applications, or host our applications there."

Central IT often caught off-guard, not by just this, but every other – this is typical within IT buying cycles, right? This happened with video conferencing, right, where every different group would use their own video conferencing and then eventually they had to consolidate and use one video conferencing, so that they had the centralized controls and that they are making sure that they had SOC's compliance and all this other stuff.

It's no different with Kubernetes. We were in the fate – Kubernetes is only a few years old, right? We're in the phase where you saw a lot of that, each line of business was doing their own thing, or each developer group was doing their own thing, and we're just getting out of that right now. We're just getting to the stage where it's starting to get back to IT and they're getting the requirements and now they're looking for a centralized way in order to manage that, but that's only happened in the past, I would say year.

[0:42:04.9] JM: When that happens, is there some ordinance, there's an ordinance from on-high that says, get your Kubernetes cluster in a way that's visible and compliant? Tell me a little bit more about how that's evolving.

[0:42:18.6] CG: Yeah. It happens in all different ways. Sometimes they're told by Gartner that they need container orchestrator, or they talk to their peers and other – the CTO has talked to their peers and there's just a order from up high of we need to do container orchestration. It literally might just be that line. We have to do this, and then someone in the organization, some architect, some enterprise architect, whoever it is to figure out exactly what the requirements are and implement this.

It happens from up top. It also happens from a line of business that could be pushing it, where they got their application running on Kubernetes, or they're using this technology and now they're the ones pushing and say, we need this in order to run our application. We would really like to have this cluster, and so they're moving in that direction.

Then it's not that – so sometimes you see within organizations that they'll have some shadow IT, whatever you want to call it, cluster that's been running, and then they'll move that onto the centralized platform, so you do see that. Other times, these shadow IT, or whatever you want to call it, this stuff that they develop in a corner, or line of business is not in a production stage. That always had to be done by central IT. They're getting the requirements now, because they want to move into that next stage.

Because these are sophisticated organizations with sophisticated determined cycles, a lot of times IT is locked down, you can't even – you can't visit, like say Gmail, or set up a Gmail within these organizations if you're on their network. It's different at startups, right? For large IT organizations, there's a very well-defined process and they're starting to go through that process now.

[SPONSOR MESSAGE]

[0:44:20.9] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost-optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW CONTINUED]

[0:46:45.1] JM: Mesosphere has a Kubernetes as a service product. I've talked to some different people who have designed Kubernetes as a service products. My sense is that there are some subjective decisions to be made in terms of how you are architecting your product and what kinds of things you want to take care of. Describe the process of managing and fleshing out the requirements of the Kubernetes as a service product.

[0:47:13.4] CG: Oh, yeah. There are a lot of knobs within Kubernetes, right? There's a lot of flags on each one of the components that you could expose or not exposed, and there's a lot of different services that you could use underneath Kubernetes, different CNI network providers, different CSI storage providers.

If you want a well-paid path for customers, you shouldn't need to make decisions on what should be the network provider for Kubernetes, what storage should they use out of the box.

Maybe they could change it out with CNI. That's very easy on the networking side. CSI is making that easier to do on the storage side. I think you talked to Gie about this in an interview.

When you're gathering requirements, especially in the Kubernetes space and especially when you're talking to large IT organizations where they're more getting pressure from up top, or maybe from developers to move to Kubernetes, they might not actually know what storage service to use, or know what the advantages of one CNI provider versus another.

You need to gather what they want to do with this technology in the end. Is it for a very something, where there's a lot of compliance issues, where they maybe need network policies? Is it something where they're going to upgrade this constantly, where they might want to use pod disruption budgets, in order to make sure that they can meet SLAs on the workloads? Are they going to do this with the line of business, where they're going to expose the entire API profile of Kubernetes, or are they going to do it more where there's CICD tools in front and the developer might not even know that there's Kubernetes underneath? There's many different use cases that organizations are using Kubernetes for. You need to hone in on exactly with what they're trying to accomplish in the end.

[0:49:16.6] JM: Those different interfaces, like container storage interface, container networking interface, etc., how do those things fit into the equation of creating a Kubernetes as a service?

[0:49:29.1] CG: Oh, because when you spin up Kubernetes, you want to have that stuff there, right? You want a well-paved road, right? You want to give them not just Kubernetes, because Kubernetes is not an island. You want to give them some network interface potentially. You want to give them some storage that they could use with Kubernetes. You want to give them image repo, so they might not have one. These larger organizations, they might just be going down that path as well.

You want all the different pieces to get a full solution, or a full product, so that they can start using it. I mean, Kubernetes is just again, the container orchestrator, or the container management. There's a lot of other things that you need around Kubernetes in order to get it into a state, where both operations feels comfortable using it and managing it and developers are comfortable, or able to use and able to actually get their workloads to it.

[0:50:25.6] JM: The container storage interface, if I recall my conversation with Gie, that's like, here's a standard interface that will allow Kubernetes to use like a, I guess, like a MySQL database. Or I can't remember the data if it's on the level block storage, or object storage.

[0:50:44.5] CG: Yeah, that's correct; the second. It's on the level of storage provider, right? Potentially, you could have a EBS plug-in, right? Or you could have NFS plug-in for CSI, or you could have even a bender, like a port works for instance, which has a storage solution for Kubernetes. That's something that you can plug in to CSI.

Eventually, there's going to be all these different options on how you do plugins, just like there is on the CNI front. A very similar model, where you could use Calico, you could use Flannel, you can use different networking overlays on Kubernetes and they're fairly easy to – actually, I shouldn't say fairly easy. It's one of the more complicated things to do in Kubernetes. The interface is well-defined, right? CNI is well defined, same thing with CSI.

[0:51:32.7] JM: That interface, that's a well-defined way of having containers communicate with each other, or having services communicate with each other?

[0:51:41.1] CG: No, having the network overlay in the case of CNI and having a having storage provider for in the case of CSI. How does Kubernetes use storage and how does it – you would have a plugin for how you would use EBS. For example, instead of using there's an option within Kubernetes of cloud provider where you could check a box and you can get a way to provision EBS, CSI would actually abstract that, or you could just do CSI and then you could use a plug-in for EBS. I hope, I'm saying the right things Gie.

[0:52:17.2] JM: That's okay. That's okay. I know. I noticed it's not your – this is not your –

[0:52:21.5] CG: Gie's expertise.

[0:52:22.7] JM: Gie's expertise. Exactly. All of this is important to Mesosphere and to DCOS and to your Kubernetes as a service, because you want to make it really easy for people to do a one-click attach something, attach a database, attach an Elasticsearch.

[0:52:41.1] CG: More than that, we want to set up everything for you, right? We want to have good defaults there, as far as networking, storage, load balancers, image repo. We want to have that, so that out of the box you're able to get up and running. Just like you would with any other cloud provider, that you have all those tools at hand, day one. Again, the idea is that that would be all packaged and that's what makes you a cloud provider. It's not just Kubernetes, it's all the other stuff that you need around Kubernetes.

[0:53:13.7] JM: From an engineering point of view, that is like, I guess it's end to one. You just have to be able to integrate, or be able to allow in different use cases, but there still are a lot of use cases. From the product management point of view, how do you manage all those different things? Do you just have a big Trello board, where you're just gradually moving each of these requirements across it as you gradually make progress on them?

[0:53:35.7] CG: Well, there's some big common requirements, like you need some network overlay right to start with, or you need storage and you need a load balancer. Those are the big ones, right? Those should be an example of the image repo, you need the same defaults, like harbor is a good one, where it is a – that's the image repo that was just donated to the CNCF. It has a good image repo, it has a hell museum, has a lot of great features. That seems like a good same default to have, right? To have a harbor repo.

You might walk into an organization and they're using Nexus, or they're using something else. You want to be able to enable that as well. You want to give them something, if they have – if a customer comes to you and they have nothing, you want to be able to say, "Well, here you go. Here is everything ready to go." If they are very opinionated on the network provider, or storage, or the image repo, you want to be able to have that pluggable, or be able to use what they currently have.

In those cases, if they have an image repo, it might be something where some well-known how to integrate that, or how do use that with Kubernetes, and so it might be something that they already know how to do is hopefully if they have it. Or it might be something where professionals, or if it's something where they don't know how to do. It might be something that where professional services could step in and actually help them out. You can't really plan for, or you can't have so many options that you plan for every eventuality, for every piece of technology that's out there.

[0:55:12.0] JM: Okay. Well, to wrap-up, are there any changes that you see to this ecosystem on the horizon, like maybe things that you're seeing ahead of most of the people in the market, just because of by virtue of where you are? What do you think are the big things that are less talked about than they should be, that are coming up around the pike?

[0:55:31.2] CG: Yeah. One big thing is obviously Kubernetes is the dominant container management, container orchestrator solution. At this point, it's fairly heavyweight, right? It does a lot. It's very generalized. If you want to do one container orchestrator, it's the container orchestrator to do. If you wanted to run just generalized workloads, I guess it has the most use cases that it could probably check the box.

If you're just – if you're a developer or organization, maybe you don't want all of that in order to just run something simple. Maybe you just need functions as a service in order to just do a quick job and/or connect to service, or something of that nature. You don't actually want to deal with, “Oh, now I have to make a container, or an image and do all that.”

I do think that's – I was talking to Kelsey Hightower the other day. There’s different tools, right? It's not like one's going to supplant the other, but I do feel that just like you had heavy weight application servers, we have this generalized container orchestrating container management. I do feel that once we settle on then and once we move forward, that there's going to be stuff out there where – we've run into it today, right? We've run into customers where they're just running maybe Gitlab, or something like that. Do you really need to set up an entire Kubernetes cluster in order to do that, or maybe just a few tools? Those questions.

Or maybe that you just want to have something where it could be satisfied by it, just a function? Do you actually need the set up Kubernetes and then entire function service just for the function part of this? In those cases, I think going forward will have a variety of different tools that will be used for different things. We are going to hone in a lot more on matching use cases to the particular technologies that they're good for.

[0:57:27.4] JM: Very interesting. Yeah, I agree with all that. Okay, well Chris, thanks for coming on the show. It's been really interesting talking to you. I look forward to our next conversation.

[0:57:35.5] CG: Good talking to you. Thank you.

[END OF INTERVIEW]

[0:57:40.4] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

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

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We’re proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]