EPISODE 504

[INTRODUCTION]

[0:00:00.6] JM: Welcome to Software Engineering Daily. Today's episode fits nicely into some of the themes that we have covered recently; Cloud Foundry, Kubernetes and the changing landscape of managed services. Sean McKenna works on all three of these things at Microsoft.

We spent much of our time discussing the use cases of container instances versus Kubernetes. Container instances are individual managed containers, so you could spit up an application within a container instance without having to deal with the Kubernetes control plane. Container instances might be described as serverless containers since you do not have to program against the underlying VM at all, and this begs the question; why would you want to use a managed Kubernetes service if you could just use individual managed containers?

Shawn explores this question and gives his thoughts on where this ecosystem is headed, but we, by no means, reached decisive answers. It's going to be an ongoing discussion as these different options evolve and we see what people use them for.

Full disclosure; Microsoft, which Shawn works that is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:01:28.2] JM: Your company needs to build a new app, but you don’t have the spare engineering resources. There are some technical people in your company who have time to build apps, but they’re not engineers. They don’t know JavaScript or iOS or android, that’s where OutSystems comes in. OutSystems is a platform for building low code apps. As an enterprise grows, it needs more and more apps to support different types of customers and internal employee use cases.

Do you need to build an app for inventory management? Does your bank need a simple mobile app for mobile banking transactions? Do you need an app for visualizing your customer data? OutSystems has everything that you need to build, release and update your apps without needing an expert engineer. If you are an engineer, you will be massively productive with OutSystems.

Find out how to get started with low code apps today at outsystems.com/sedaily. There are videos showing how to use the OutSystems development platform and testimonials from enterprises like FICO, Mercedes Benz and Safeway.

I love to see new people exposed to software engineering. That's exactly what OutSystems does. OutSystems enables you to quickly build web and mobile applications whether you are an engineer or not.

Check out how to build low code apps by going to outsystems.com/sedaily. Thank you to OutSystems for being a new sponsor of Software Engineering Daily, and you're building something that's really cool and very much needed in the world.

Thank you, OutSystems.

[INTERVIEW]

[0:03:19.6] JM: Sean McKenna is a principal program manager at Microsoft. Sean, welcome to Software Engineering Daily.

[0:03:24.6] SM: Thanks, Jeff. Good to be here.

[0:03:26.6] JM: Yes, it's great to have you. You've been at Microsoft since the beginning of Azure. Describe the early days of when Microsoft started building a cloud business.

[0:03:39.2] SM: I've actually only joined Azure in the last couple of years, but when I was working elsewhere at Microsoft, I was a user of Azure and so I sort of have seen the evolution of the platform. It's interesting, we sort of started out, Microsoft being a platform company, thinking of the cloud in terms of developer platforms, and so we started with what we now call cloud services, which is effectively a platform as a service layer where you could create your application and it would automatically spin up the underlying infrastructure for you and create the necessary networking components and low balancers and all of that and it would all happen sort of behind-the-scenes, which was a great developer experience for a particular customer profile. In those days it was really .NET developers using Visual Studio. If you fell into that camp, you had a great sort of end-to-end experience.

What we found overtime, what the Azure team found overtime was, “Hey, there’s a much broader set of use cases that people are looking to tackle with the cloud,” and that's where Amazon in particular had really driven this revolution around just infrastructure as a service. So having the ability to just use raw virtual machines and connect those up to other basic components that the customers might've been used to running in an on-premises environment, and so that's where we went ahead and introduced Azure's IaaS services and that's really been what's driven it to take off to date, although now we’re sort of moving into the next phase and actually interestingly moving back up the stack with platform layers. So it's been kind of an interesting full circle evolution for Azure.

[0:05:19.9] JM: What’s funny, that sounds similar to what Google's approach. Google started out with App Engine, and App Engine was beloved by some people, but was probably a little too opinionated for the vast majority of use cases. It sounds like that's quite similar to what you had going on.

[0:05:40.1] SM: Exactly. Yeah, I think both of those were in some ways sort of ahead of their time, that they were too high of an abstraction for what a lot of customers were looking to do at the time, which was just, “Hey, I want to pull the plug on my data center. I want to be able to move my existing applications,” what we referred to as sort of lift and shift, “take my existing applications that are running BMs, around bare metal in my on-premises environment. Just move them into the clouds. That's all I want to do. I don't want to rewrite my applications. I don’t want to re-platforms on to your specific offering. I just want to kind of get out of get out of my data center.” That was sort of one part of it, and then the other was, of course, there were specific things you could do in those environments. There were specific frameworks and languages that were supported, and so if you didn't fit into that particular bucket, then there wasn't really a great solution for you. So that's where these IaaS offerings as just kind of base building blocks for building applications, which are in most cases a lot harder for developers to build applications on, but at least, really, anything you wanted to build, you could do with that base infrastructure.

[0:06:51.1] JM: You work on a number of different products at Azure. One of them is Cloud Foundry. You also work on some Kubernetes related products and these new container instances, and I've been doing a number of shows recently about both Cloud Foundry and Kubernetes and I'm trying to get an understanding of the business landscape, how enterprises are choosing between the different platform as a service offerings, and it seems like for some period of time, really, they just opted for Cloud Foundry, and that was pretty much the only game in town. If you're a bank and you want to be on a platform-like experience in the cloud or if you're a telco, these are the types of customers that I've seen the most use Cloud Foundry. Are you getting a sense that Kubernetes is appealing to them, that they're migrating away from Cloud Foundry towards Kubernetes or is it a complementary sort of thing? How has Kubernetes affected that calculus for these big enterprises?

[0:08:06.2] SM: Yeah. Certainly, what we've seen is a lot of the big enterprises have historically liked Cloud Foundry for, first of all, the portability across multiple environments, but also just having sort of an end-to-end opinionated developer experience that everything is sort of available out-of-the-box and you don't have to think about how to set up going from writing code to having it deployed in a cloud environment to doing day two operations, thinking about logging and upgrades and managing infrastructure. That's been something that that customer profile has gravitated towards.

I think with the rise of containers, there's been a lot of interest in Kubernetes. It's kind of the de facto container orchestrator, and in fact, now, Cloud foundry has incorporated Kubernetes into sort of its ecosystem. So now Cloud Foundry is actually an umbrella term for both the sort of application run time, which is what was previously known as Cloud Foundry and then the container runtime, which is actually a distribution of Kubernetes.

They're looking to bring those together under one roof, because there's a lot of customers that are looking for some mixture of the two, where you might want to have your custom built applications where you actually have developers inside the organization writing custom software and you want to have them be able to deploy that into a cloud environment without having to think about how do I actually go about containerizing this and managing it as a container. Just have that sort of whole workflow managed by the platform, and then there're other cases where you potentially want to take off-the-shelf software, which is now largely distributed in containers and be able to deploy that into a Kubernetes environments. Those are kind of potentially working in concert with each other.

The other thing that we are seeing is that the Kubernetes ecosystem is maturing and kind of moving up the stack. I think overtime we'll see the full Kubernetes sort of full sort of CNCF stack actually provide potentially a platform experience that is more in line with what people have been able to historically get with something like Cloud Foundry. It's a question of whether it ever gets to the level of opinionation that something like Cloud Foundry has, but I think it is starting to move a little bit further up that stack, such the difference in the level of technical expertise that you need to be able to run Kubernetes versus Cloud foundry may actually reduce.

[0:10:46.9] JM: Yeah. I think you're referring to the Kubernetes platforms like OpenShift or Platform nine or rancher. There's a bunch of these different platform as a service that are just built on top of Kubernetes.

[0:11:02.2] SM: Yeah, that's one part of it, is sort of the commercial distributions of Kubernetes. There's also sort of all the pieces that are starting to fill in around Kubernetes. One of the things that Brendon Burns, who's one of the founders of Kubernetes and now runs container team that Azure, has always said is, "Kubernetes is not meant to be the end of the story. It was meant to be a sort of base building block that other things would get built on top of," and we're starting to see that now with the community rallying around things like Prometheus for logging, Istio for service mesh. There're these pieces that are getting plugged in together that when you put them together, they're not parts of the Kubernetes core necessarily, but if everybody rallies around those being, the sort of de facto standards in each of those particular areas, then you start to build a stack that is more of an end-to-end story. Whereas Kubernetes by itself, it's the container hosting and orchestration, but there's a bunch of exercises left of the reader to actually build applications and operate them on day two. So we're sort of still seeing that maturation process happening in that community, but I think there is a lot of consensus building around some of those tools.

[0:12:25.3] JM: Help me understand the mindset of the telcos or the banks or the oil companies that would be making this kind of selection. Which pads am I going to start to move my infrastructure to? How much of these typical enterprises, how many of them are what percentage of their infrastructure have they moved to a PaaS Is there a lot of migration that is still entirely on legacy systems, like do you have a percentage in mind?

[0:13:02.2] SM: That really depends on the organization. Some have been fairly aggressive. I would say most of them are still running a minority of their workloads in some of these environments. They're usually looking for low hanging fruits that they can start to migrate over as kind of test cases, and then slowly expand that through the organizations. So I don't know, percentage-wise, maybe 10%, 20%, maybe a little bit more.

[0:13:29.5] JM: 10%, 20%, they have done the migration and they still have like 80% left.

[0:13:33.6] SM: Yeah, that's what see — With a fair number of customers, if you look at something like Cloud Foundry, for example, typically what customers will do is they'll get a Cloud Foundry environment set up, they'll cherry pick a couple of applications that they want, either existing applications that they want to move into that environment, or often if they're doing greenfield development, they want to build new applications with the sort of micro-service oriented architecture and they'll target those to move into Cloud Foundry and kind of start to build the organizational muscle both in terms of development as well as operations using that environment and kind of make everybody comfortable with that model.

Then after they've got that up and running and it's been successful, they'll go move on to the next phase and sort of slowly move more and more applications over. That's typically what we see. It's not that they're going to do a big bang, try to move everything over and in three months, six months, that kind of thing. People want to feel comfortable with working with these new platforms. In a lot of cases, they're changing from an IT model that they've been using for 10, 15, 20 years. So there's naturally some hesitation there to moving to an entirely new way of building and operating software.

[0:14:50.6] JM: When you're talking to these customers — Since you're in charge of the Cloud Foundry at Azure, but you're also heavily involved with the Kubernetes people, are you starting to talk to vendors where you're not sure how to advise them or are you helping them pick between like Kubernetes versus Cloud Foundry? How are those conversations going?

[0:15:12.3] SM: Yeah, we definitely get a lot of customers that are asking for guidance on sort of which model to adopt. At this point, typically it comes down to sort of the maturity of the IT organization within in that enterprise. So if a company is, like I was saying before, looking for kind of an opinionated end-to-end platform for doing custom in-house development of applications in a cloud native way, Cloud Foundry is probably the way that they want to go today, because it does give them that sort of end-to-end story. They don't need to think about stitching together a bunch of different pieces around Kubernetes.

Now, that's very much up a point in time kind of conversation and it may change over time, but typically, today, if somebody's looking for that in opinionated end-to-end story, it's going to be one of those higher-level PaaS layer, so Cloud Foundry or sort of the Microsoft landscape, something like service fabric, which is more of our .NET Windows oriented platform.

for customers who are potentially a little bit more advanced, or at least trying to build up that muscle and are specifically attracted to some of the benefits of containers, so sort of the idea of immutable infrastructure, being able to create those application packages that are going to run consistently across environments and are willing to do a little bit more of the legwork in terms of actually building out a platform by pulling together those additional layers, some of which I was referring to earlier in terms of logging and service mesh and that sort of thing. Then Kubernetes community is growing pretty rapidly and there're a lot of resources available there to help those customers, but it is something where they're going to have to do a little bit more work as it stands today.

[SPONSOR MESSAGE]

[0:17:20.1] 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.

Check out the Azure Container Service at aka.ms/acs. That's aka.ms/acs, and the link is in the show notes. Thank you to Azure Container Service for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:18:48.4] JM: You work on several other projects, like I said. One project is the virtual-kubelet, which is a system for connecting Kubernetes to other APIs. This is like if you are running a Kubernetes cluster and you want to connect to other cloud service APIs through a virtual-kubelet, you can do that. Explain what a virtual-kubelet is.

[0:19:12.2] SM: Yeah. A little bit of history. So in July of last year we launched a service called Azure Container Instances, and the easiest way to think of that is basically Kubernetes pods as a service. So the ideas you can run individual containers or what we call container groups, which is effectively a pods on multiple containers that share a lifecycle and are cohosted on the same machine and have access to each other via local network and so on.

You can create those in the Azure cloud without having to provision any underlying infrastructure first. Tight, if you want to run pods in Kubernetes, you set up Kubernetes, you're creating a cluster of BMs that you then have to manage at some level. So the idea behind ACI is I can just deploy pods or container groups directly into Azure without having to do that.

As part of the design of ACI. one of the things that we were kind of wrestling with is, okay, when we launched this service, which is your allowing you to run containers without underlying infrastructure, where we do per second billing, there's a number of different things that are quite attractive about it. Naturally, we're going to start to get questions about, "Okay. How do I scale out? How to I create a number of replicas of these containers? How do I do upgrades to them? How do I manage availability?" Kind of all of the things that you think about container orchestrator doing, so something like Kubernetes.

The decision we made was rather than go and add all of those capabilities to ACI, to that core product, which is ultimately always going to be something that is specific to Azure, it's a core part of the Azure infrastructure, that we would work sort of in conjunction with Kubernetes and allow Kubernetes to be the orchestration layer. It's a fairly mature orchestration API. There's a lot of activity in that community, and so we want to actually be able to bring those two things together so you can get the best of ACI with the best of orchestration in Kubernetes.

So we launched in connection with the launch of ACI, we've created a project called the ACI connector, and the idea behind that was to allow a Kubernetes cluster to have this kind of virtual node that would allow you to schedule pods into ACI in addition to scheduling them on BMs. When we built that project, it was ACI specific. So we had literally just created one that would only enable you to deploy pods into ACI.

But after we launched it, there was a bunch of interest from other similar providers. So Hyper.sh, for example, which is one of the leaders in this kind of serverless container market actually went and built sort of an equivalent connector that would allow customers to target Hyper instead of ACI, and so we saw this kind of interesting in this model of a hybrid between traditional Kubernetes s cluster and these sort of serverless containers or pods as service offerings.

So we decided to go ahead and sort of up level our ACI connector project into something that would be a pluggable framework so that different providers could actually plug into that model. So that's what we launched at Cube-Con, was the open-source, sort of upstream project for virtual-kubelet along with a provider for ACI, and we're working with hyper there in the process of adding a provider for hyper, and we've heard from a number of other companies both at Cube-Con and subsequently that are interested in adding those providers. That's basically the idea, is to allow you to have a kubelet, which is not a traditional kubelet in the sense that it's not an agent running on a single VM, but is representing kind of an agent or a node that has infinite capacity.

[0:23:10.6] JM: What's an example for how I would use the virtual-kubelet?

[0:23:14.0] SM: Yeah. We expect there's kind of eventually going to be two primary ways that it gets used. The first one is the ability to spill over capacity into something like ACI. If you imagine, today, if you're running a Kubernetes cluster, you've got a set of VMs. Part of the value that you're getting from using container orchestrator is you're getting a pretty high utilization, because you've got this sophisticated scheduler that can map the workloads that you were looking to schedule on to the set of resources that you have and make sure that you're packing those in as efficiently as possible. But typically, you're still going to leave a fair bit of capacity on each of those VMs for the potential that you might get; an external spike of traffic if you're, say, a retail website that needs to be prepared for some flash sale, or if you have a like a regular batch job that needs to run at the end of the week, the end of the month, or maybe you're running Jenkins inside of your cluster and you've got kind of spiky traffic as a result of when you've got builds the need to run. So you keep that additional capacity around for when those workloads are going to come in, but all of the time when those things are not happening, you effectively got wasted resources. So you're paying for something that you're not using.

The idea behind the virtual-kubelet and things like ACI is you can connect that into that cluster and use that as effectively like your overdraft protection if when you need that additional capacity. So that allows you to really have your VM-based cluster, the environment for running your sort of standard workload, your average workload, drive up that utilization to 80%, 90%, and if you need it, have that additional capacity come via the virtual-kubelet from something like ACI. That's sort of one scenario that we're looking at.

The other one is we think eventually that a lot of customers are just going to want to run effectively, like a serverless Kubernetes where you can have Kubernetes scheduling pods into infrastructure where you don't necessarily see all of the individual VMs. So in that case you would purely be using a Kubernetes hosted control plane, such as the Azure container service that we offer, or others have equivalent offerings, but then have the actual hosting environment be managed by, like in this case, Azure, where the infrastructure would be hidden from the customer. You can just focus on building and managing your containers, which is quite a nice model, because if you think about it today purely in terms of scaling, there's always this kind of two level scaling problem, where when you need to add additional capacity to your application, your containerized application running in Kubernetes, you would initially scale out the number of pods. You'd use the horizontal pod auto scaler to add additional capacity that way, but eventually you're going to run up against the limits of the infrastructures. So how many VMs you have to actually schedule those pods? So you're always having to do this sort of back-and-forth dance to make sure you have sufficient infrastructure to manage the scaling in and out of those pods. Whereas if you move to a model where it's all kind of managed for you, you can purely think about it at the container or the pod level.

[0:26:45.0] JM: So that world in the future with the serverless Kubernetes idea, is that the same thing as the container instances?

[0:26:56.7] SM: Container instances is the hosting environment for the individual pods. You can think of it as — When I talk about sort of the serverless Kubernetes, it is the Kubernetes API. So thinking in terms of Kubernetes deployments and services and pods and all of the Kubernetes constructs, but the actual runtime hosting environment would be something like ACI. I would still be using kubectl and the Kubernetes API, but rather than those pods landing on VMs that I have to manage and think about scaling and all of that, they would just land in — You can sort of think of it as this sort of PaaS in the sense that the infrastructure is hidden, but you're still managing everything to Kubernetes. You can do — kubectl get pods and see all the pods that are running. You could rolling upgrades with deployments. All the things that you would expect to be able to do with Kubernetes, it's just that rather than those pods running in VMs, they're running in this kind of hidden infrastructure.

[0:28:01.8] JM: So in that world, what differs from that world where you have this serverless control plane that's managing Azure container instances versus a managed Kubernetes offering?

[0:28:18.5] SM: Yeah. So when we talk a managed Kubernetes today, and this is the case for other managed Kubernetes offerings as well, is the management piece is effectively the master nodes. In a Kubernetes cluster, you've got the master nodes and then the agents, where the masters are running the API server, the scheduler, etcd, all of the kind of the brains of the cluster. Managed Kubernetes services take those components and move them into the IaaS provider's infrastructure. So in our case, move them into infrastructure that we manage in Azure, but you still have VMs in your subscription, in your account that are the target of those deployments. Now, those VM's may themselves be managed at some level, but you are still looking at the VM as the unit of billing, as the unit of scaling, and to some degree, as something that you need to manage.

If you move to a model that is sort of a purely serverless Kubernetes kind of model, you still have the management piece that runs inside the IaaS provider's infrastructure. So that part doesn't change. What changes is that the target of your deployment is not the set of VMs in your subscription, but is something like ACI So you would, in terms of the resources you would see in an Azure, for example, you would just see the containers or just see the pods as a first class IaaS resource, rather than if you set up a Kubernetes cluster today in Azure and go into the Azure portal, for example, or use CLI, there's no way for me to see the set of pods that I've deployed, because those are sort of hidden inside of Kubernetes in a —

[0:30:10.9] JM: That's even on AKS.

[0:30:13.7] SM: Yes. Those pods are effectively living inside of VM's, and so from an Azure perspective, all I see are VMs, right? I'm going to have to go through Kubernetes to see the actual pods. In serverless world, I'd be able to go through the Kubernetes API and the kubectl, cli, and typically that's what I would do, but from an IaaS has perspective, I would see my pods rather than any VMs, and that would be my unit of billing and my unit of scale. So now the infrastructure, the underlying VMs just get hidden away, which we sort of think of as just the next natural step in IaaS, right? With IaaS today, we will give you virtual machines that sit on top of physical infrastructure that you don't ever see.

In the same way here, we're giving you containers as an IaaS primitives that sit on top of infrastructure that you don't see. It's kind of just a natural evolution of virtualization and of infrastructure as a service that we think makes sense in the context of something like Kubernetes.

[0:31:22.3] JM: What surprises me is that this abstraction of the container instance was not available like two years ago. When Docker started getting popular, there was not a way to buy and provision a single container. There was Heroku, which you can get Dynos, which I think a Dyno always was a container. Why is that? Why didn't we have the major cloud providers offering the single container instance as a purchasable abstraction?

[0:31:55.4] SM: I think it was because a lot of — The container orchestration platforms we're looking to operate with the lowest common denominator kind of environment. Azure container instances is something that is effectively a service that we provide in Azure that you wouldn't be able to pick up and take to you on-prem environment, for example. Whereas the orchestration platforms wanted to be able to run either directly on physical hosts, around VMs in your on-prem environment or in public cloud providers, and so make sure that those work, those APIs work everywhere. So that was the natural place to start, but I think what we're going to see now is at least every cloud provider will likely have some kind of offering that is purely container-oriented. I think it's just a natural evolution. Everybody wanted to make sure that their offering was working in all environments, but now we'll start to see people sort of moving up the stack.

[0:33:01.6] JM: When that is unavailable, purchasing and use deployment model of this serverless Kubernetes or serverless containers, container instances that you can just purchase one by one. It's a very appealing abstraction for many applications that I can think of. Where does that leave something like your managed container service, that AKS? Like who will be using that in four or five years? Do you think there'll be more platforms that will evolve that will develop that people will deploy their own platforms on top of AKS and then sell whatever is the higher level abstraction that they build on top of that?

[0:33:49.6] SM: We certainly expect that to happen, but I don't think — The whole idea of the virtual-kubelet project is that we think things like ACI and AKS are complementary rather than competitive. So we think there's a set of use cases for ACI standalone that we see people doing today where they're spinning up CI jobs or they're doing kind of simple task automation or very simple applications, but as soon as they move into more complex applications or they want to have some of the things I was mentioning earlier in terms of scaling and rolling upgrade and service discovery and all of the kind of the things that orchestration APIs provide, that's when they want to move over to something like Kubernetes, in our case, in the context of AKS, but they still might want to have the actual hosting not in VMs, but just in managed infrastructure.

We think there's this kind of marrying of the capabilities of an orchestration API with the hosting model, hosting and billing model of ACI. You might be building a multiservice, micro-service-oriented application and you're deploying that as lie a Kubernetes deployment or you're using a Helm chart or what have you, so all the things you would typically do with Kubernetes, but have those pods ultimately be deployed in ACI and paid by the second for however long they're running and not have to think about scaling out the set of VMs. We really think that long term, those things continue to be complementary, where we're just swapping out the core hosting infrastructure of VMs to being this sort of higher level abstraction.

[0:35:39.7] JM: In the build out of the managed, the container instances, were there any specific technical challenges to work out that come to mind?

[0:35:51.8] SM: Yeah. One of the biggest things is we are providing full hypervisor isolation for these containers running in ACI, and so making sure that we could run them in managed infrastructure that we provide, which is sort of naturally multitenant, but have them be hypervisor-bounded but still give you the expected sort of startup times that you would come to expect from containers. We've been working with both the Windows team in terms of hyper-V containers and being able to take advantage of some of the capabilities that they've added there and some of the performance improvements, and then we've also been working on the Linux side with Intel Clear Containers or what's now been rebranded to Kata Containers where the idea is to have kind of the best of both worlds of what you have with a traditional VM where you have strong isolation between environments running on the same physical infrastructure and containers where you have this much smaller images and much faster startup times, and so making sure that we're able to provide that hypervisor isolation was one of the big challenges and something we had the kind of go down a number of paths to make sure that in terms of network access and storage and kind of how we do logging, that we had that clear isolation, because that's something that customers are looking for. If they're going to run these applications in this hidden infrastructure, they need to feel confident that we are actually isolating those environments and providing them equivalent level of isolation that they would get from a VM.

[SPONSOR MESSAGE]

[0:37:48.8] JM: If you are building a product for software engineers or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an email, jeff@softwareengineeringdaily.com if you're interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers because I talked to them all the time. I hear from CTOs, CEOs, directors of engineering who listen to the show regularly. I also hear about many newer hungry software engineers who are looking to level up quickly and prove themselves, and to find out more about sponsoring the show, you can send me an email or tell your marketing director to send me an email, jeff@softwareengineering.com.

If you're listening to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company. So send me an email at jeff@softwarengineeringdaily.com.

Thank you.

[INTERVIEW CONTINUED]

[0:39:16.5] JM: As the world moves towards this world where people are using Kubernetes more and more, and Kubernetes is on all the cloud providers, it's a managed service that's present everywhere, how does that change the calculus of a cloud provider? What's in the future for a cloud provider strategy?

[0:39:38.3] SM: Yeah. So we kind of see it as, eventually, managed Kubernetes services are just going to be table stakes and then it's not going to be something that any individual cloud provider can differentiate on n its own. So we need to provide additional value. We need to make it attractive for developers or organizations that want to build on top of Kubernetes to come to Azure. So there's obviously, in our case, the main benefits of our IaaS that we typically talk about are regional coverage. We've got the largest number of regions worldwide. So for large enterprise customers who are looking to build out global deployments and potentially deploy applications into mainland China or into Germany and deal with data sovereignty or privacy regulations in those countries, we offer a great model for doing that.

We also expect that one of the things that we will do is really build on our legacy of building great tools for developers and really democratizing technologies. So we're making a lot of investments in providing great tooling for Kubernetes. So we are now the stewards as a result of our acquisition of [inaudible 0:40:57.6]. We're not the stewards of the Helm Project. That same team has built out draft, which is making the developer inner loop for targeting Kubernetes much easier. Tools like Brigade, which is this event-driven pipelines in Kubernetes and [inaudible 0:41:15.2], which allows you to do sort of CI/CD kind of models. There's a number different tools there. Building integrations with things like BS code. So really trying to provide that end-to-end developer experience where we will naturally make it the easiest way or the easiest environment to target to be Azure, and so that something natural for developers who are targeting Kubernetes to deploy their applications into Azure. We don't expect to really differentiate to a large extent on the core managed Kubernetes services. It's going to be all the things that come around it.

[0:41:51.2] JM: Do you think that like the clouds are going to become increasingly differentiated because everybody is on Kubernetes? Because, I mean, it feels like several years of the core offerings were fairly similar from cloud to cloud, but it feels like now things are starting to get more differentiated. You're starting to see more exotic services across the different cloud providers. Do you think they'll continue?

[0:42:18.9] SM: Yeah, I think there's going to be a mixture. In the context of Kubernetes, like I said, I think at this point now we have announcements or running services from the three major cloud providers around managed Kubernetes services. So that will be fairly consistent. In fact CNCF has a Kubernetes certification program specifically to ensure that these different distributions and these different services are consistent to allow customers to make their applications portable.

Yeah, it's going to be up to the cloud providers to determine how they want to differentiate beyond that. I think we're going to see a mixture of consistent set of services. If you think about things like managed database services, we have — In Azure, we have a managed SQL Server and as well as a managed MySQL and managed PostgreS. So those are — At least in the case of MySQL and PostgreS, those are standard open APIs that I expect either exist or will be services, equivalent services from other cloud providers. That's something you can take advantage of if you want to have your application be portable.

On the other end of the spectrum, we're going to build differentiated services. So things like CosmosDB, which is our globally distributed NoSQL database, which itself has a set of open APIs that sit on top of it but has differentiated capabilities. So that's going to be, I think, the decision of individual customers where they want to go deep with a particular cloud provider because they see that there's differentiated value there, versus trying to keep their applications and their operational model consistent so that they can move them between environments. We see kind of a mixture of those two types of customers with different approaches.

[0:44:03.8] JM: So let's say I'm a developer that wants to take advantage of exotic services on all the different cloud providers, does that mean that I will more than likely end up's spinning up a Kubernetes cluster on those different cloud providers as well so that I can also have bespoke services on those — Running in my own containers on those cloud providers that are interfacing with the exotic services, like maybe I've got a Kubernetes cluster on Azure that is interfacing with CosmosDB, because I want that, and then I've got a Kubernetes cluster on Amazon, because I want to interface with some Amazon managed services. Do you think that's a viable model or is it more likely that we'll just have single Kubernetes cluster on one of these cloud providers, and if you want to interface with an exotic service, you just hit their Rest API? What's your vision for that?

[0:44:59.4] SM: I think, typically, we'll see customers deploying Kubernetes clusters in the cloud provider where they want to connect to those services. You can employ pretty small clusters and you have the consistency and the mutability of containers such that those containers running in those different environments will operate the same way, and so you don't necessarily need to run everything in one environment. It also typically makes things easier in terms of networking and security to have those services running in the same environment, and so I expect that there's going to be that kind of model of you run a set of applications and it may be the same application running in different cloud environments, or more likely, I think from what we've seen with customers, is they kind of choose the types of applications that makes sense to run in different cloud providers depending on the services that they want to take advantage of or things like I mentioned with regional coverage, where if they've got some application that is specific to, say, the German market or the Chinese market, that they might run those in in Azure, because we have coverage in those regions. Yeah, I think it's going to be mostly that you'll run your stateless applications in that cloud provider and connect to the cloud provider services

[0:46:24.4] JM: This is one thing that I think is understated about the idea of multi-cloud, is that people are going multi-cloud sometimes to take advantage of the upside of being on a different cloud, and I think the common narrative or what I've heard the most is that you want to be on multi-cloud, because it's like you want to be ready to lift and shift or you want to be disaster recovery tolerant. Maybe that's true, but honestly I've seen much more of the former rather than the latter.

[0:46:56.3] SM: Yeah, absolutely. I mean, we do here customers who are thinking about multi-cloud in the sense of they're going to be doing kind of like per second arbitrage of moving applications between different cloud providers based on cost and things like that, but I think that's — In a lot of cases, that kind of model is a bit of a pipe dream, and in any case, it's going to be pretty small.

[0:47:20.3] JM: Although that was everybody's pipe dream. When all these cloud provider started standing up, people were like, "Oh! Who's going to build the market between them?" That never really happened, because everything went to zero.

[0:47:33.5] SM: No. Yeah, and I mean, by virtue of the competition being so fierce, the costs don't typically get too far out of alignment, at least not enough to warrant trying to really make significant return from moving things between them. Yeah, I agree. I think most of the time, customers are choosing — If they are indeed doing multi-cloud, it's because they see different values from the different cloud providers, and that's great. I think having a world in which there's a set of things that are consistent, we're all going to provide VM's, we're all going to provide some standards-based data services. Potentially, we're all going to provide containers as a service at some point, but then we're also going to go and do differentiated things and we're going to do interesting innovations and we'll see kind of what things the market reacts to. I think that's kind of the perfect competitive environment, frankly.

[0:48:32.2] JM: Can you give me a little bit of insight into how product development works at Azure, like that on the cloud stuff that you work on, just because I do a lot of shows with — Products that are easier to think about and how you would structure an engineering team around, like shows where I'm doing something like about a dating app, like I did some shows about Tinder and some shows about thumbtack, which is a marketplace, and it's kind of easy to understand, "Okay. There are some operational people. There are some backend engineers. There are some frontend engineers." Complicated services, but it's kind of straightforward when you think about it. Compared to developing services on top of a cloud provider, my understanding of how to manage that kind of breaks down. Maybe you can give me some insight on how management and strategy works for the teams that you work on.

[0:49:26.7] SM: Yeah. Organizationally, we have teams in terms of kind of the core infrastructural components, so compute storage networking. We have teams that kind of own those as base infrastructure layers, and then they're sort of further subdivided based on largely sort of around the architecture. They have to work together to provide that base set of capabilities, and then on top of that, building higher-level services like ACI. That's actually more of a traditional kind of product team where we've got myself and a couple of other folks as product management and then we have an engineering team of about 10 people at this point where we're just managing a backlog of items based on customer demand and kind of where we want to take the service, and we do sort of monthly planning and sprints and really drive that service kind of end-to-end. It sort of depends on the level of the stack that you're at how the teams operate.

[0:50:33.8] JM: Okay, makes sense. Last question, I did a show with Brendan Burns and also a show Joe Beda, and they both said something similar that really struck me, which is they articulated this idea that there's going to be this different future where people can write — Well, at least this is the way Brendon put it is, write their own smaller software teams, write and distribute proprietary software using Kubernetes as that distribution layer. So I could sell —

Maybe I create my own database and I sell it on Kubernetes through helmet, which is that's something that doesn't really exist in software today. If you buy software, you're mostly buying it as a subscription. You're buying software as a service, but you can imagine just paying a flat fee for a binary and then you're paying for the ongoing cost of running it on a cloud provider which would be de minimis compared to the cost of subscription cost of a SaaS company. It sounds like you agree with that potential future. Do you have any ideas for what the roadmap looks like to getting there?

[0:51:47.6] SM: Yeah. So that's something that we're looking at currently in the context of the Azure marketplace. It's still fairly early days I would say. We don't have concrete timelines for when we might have something there, but yeah, we have a pretty rich marketplace today for software being distributed in VMs, but what we're seeing is a lot of ISPs, being interested in doing distribution via containers. So I think there's a lot of opportunity there for thinking about different licensing models and different pricing models for ISV software that might be running alongside the domain application in a Kubernetes cluster that the customer manages. Yeah, definitely a huge area of opportunity and one where I think we can provide a lot of monetization opportunities for ISVs. So it's pretty exciting.

[0:52:42.3] JM: Do you think that'll look like a cross-cloud marketplace? Because my understanding of the cloud marketplace as they stand today is it's a little bit fragmented. Like if I buy something in an Azure marketplace, it might be — I might not be able to buy that in the AWS marketplace or the Google marketplace and which could kind of make sense, because it's a different VM style that I would be installing it too, but with Kubernetes, it would be a little more standardized. Do you think that would create a better environment for distribution?

[0:53:12.9] SM: Yeah, this might be one of the cases where having some abstraction that sits across the cloud providers is actually beneficial. So having a common pricing and licensing model that can be sort of arm's length from any individual environment and maybe it goes beyond cloud providers. Maybe you can run that ISV software in a Kubernetes environment that you're running in your own on-premises data center.

Yeah, adoption of Kubernetes by the community broadly I think opens up a whole bunch of these opportunities, because you now do you have that consistent higher level abstraction that now you can go and build on top of. This is just one of many opportunities I think for new ways of distributing software to happen.

[0:54:02.7] JM: Okay, Sean. Well, great talking to you. I really enjoyed the conversation, and lots of fascinating insights on the business implications and the technical implications of Kubernetes. So thanks for coming on the show.

[0:54:13.9] SM: All right. Thanks, Jeff. I really enjoyed it.

[END OF INTERVIEW]

[0:54:17.9] JM: GoCD is an open source continuous delivery server built by ThoughtWorks. GoCD provides continuous delivery out of the box with its built-in pipelines, advanced traceability and value stream visualization. With GoCD you can easily model, orchestrate and visualize complex workflows from end-to-end. GoCD supports modern infrastructure with elastic, on-demand agents and cloud deployments. The plugin ecosystem ensures that GoCD will work well within your own unique environment.

To learn more about GoCD, visit gocd.org/sedaily. That's gocd.org/sedaily. It's free to use and there's professional support and enterprise add-ons that are available from ThoughtWorks. You can find it at gocd.org/sedaily.

If you want to hear more about GoCD and the other projects that ThoughtWorks is working on, listen back to our old episodes with the ThoughtWorks team who have built the product. You can search for ThoughtWorks on Software Engineering Daily.

Thanks to ThoughtWorks for continuing to sponsor Software Engineering Daily and for building GoCD.

[END]