EPISODE 490

[INTRODUCTION]

[0:00:00.3] JM: In the first 10 years of cloud computing, a set of technologies emerge that every software enterprise needs; continuous delivery, version control, logging, monitoring, routing, data warehousing. These tools were built into the Cloud Foundry project, a platform for application deployment and management.

As we enter the second decade of cloud computing, another new set of technologies are emerging as useful tools. Serverless functions allow for rapid scalability at a low cost. Kubernetes offers a control plane for containerized infrastructure. Reactive programming models and event sourcing make an application more responsive and simplify the interactions between teams who are sharing data sources.

The job of a cloud provider is to see new patterns in software development and offer tools to developers to help them implement those new patterns. Of course, building these tools is a huge investment. If you're a cloud provider, your customers are trusting you with the health of their application. The tool that you build has to work properly and you have to help the customers figure out how to leverage the tool and resolve any breakages.

Onsi Fakhouri is the senior VP of R&D for cloud at Pivotal, a company that provides a software and support for Spring, Cloud Foundry and several other tools. I sat down with Onsi to discuss his strategy for determining which products Pivotal chooses to build. There are a multitude of engineering and business elements that Onsi has to consider when allocating resources to a project.

Cloud Foundry is used by giant corporations like banks, telcos and automotive manufacturers. Spring is used by most enterprises that run Java, including most of the startups that I have worked at in the past. Cloud Foundry has to be able to run on premise and in the cloud providers like AWS, Google and Microsoft. Pivotal also has its own cloud, Pivotal Web Services, and all of these stakeholders have different technologies that they would like to see built. Onsi's job is to determine which ones have the highest net impact and make a decision on those and allocate resources towards them.

I interviewed Onsi at Spring One Platform, which is a conference that is organized by Pivotal who, full disclosure, is a sponsor of Software Engineering Daily. This week's episodes are all conversations from that conference, and if there's a conference that you think I should attend and do coverage at, let me know. Whether you like this format or not, I would love to get your feedback. We have some big developments coming for Software Engineering Daily in 2018 and we want to have a closer dialogue with the listeners. Please send me an email, jeff@softwareengineeringdaily.com or join our Slack channel. There is a link in the show notes.

Thanks for listening.

[SPONSOR MESSAGE]

[0:03:08.9] JM: Simplify continuous delivery GoCD, the on-premise open-source continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and you can visualize them end-to-end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.org/sedaily, find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.org/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thanks to GoCD for being a continued sponsor of Software Engineering Daily.

[INTERVIEW]

[0:04:08.5] JM: Onsi Fakhouri is the senior VP of R&D for cloud at Pivotal. Onsi, welcome to Software Engineering Daily.

[0:04:14.9] OF: Thanks. Good to be here.

[0:04:16.4] JM: You manage R&D at Pivotal, and Pivotal has a lot of products. How do you decide which products to direct resources towards?

[0:04:26.2] OF: That's a really, really good question and it's one that we've been working to improve on as we've scaled. We've been this rocketship journey going from just a handful of small teams to now over like 260 people. I want to focus on talking about Cloud Foundry here specifically. So how do you scale an agile organization? How do you scale product for an agile organization? How do you do it when you're running cross-multiple offices? How do you that? How do you do that well, and how do you — Sort of in my position in leadership, how do you do that without overly centralizing control, but actually decentralizing authority and decision-making? What we really, really want to do is empower and enable our product managers who are running our various components to our various component teams to really feel empowered to make the right decisions around what to prioritize and when and how to move the product forward.

I think we've had a lot of success there and it's so interesting that you have to really organize your teams in a way that promotes success. We have some teams that just totally own a piece of the product and they can move quickly and they can make their own prioritization decisions, and then the way that staffing works is they will bubble up, "Hey, here's the business case for why we need more," or the teams health demands that we have another pair or another two pairs and then that feeds into sort of the allocation system where we're just trying to embrace like this marketplace mentality of flowing resources to where the impact can be highest, right?

What's been really an interesting and fun challenge is how do you take teams that are organized around components, because they sort of have to be, because each of our pieces of Cloud Foundry is so complex that you need engineers that have a depth of experience who stick with it, who go really deep. We do everything from containerization where we find ourselves debugging the kernel, because of like a production issue, right? All the way to like streaming logs through the system, to handling routing of traffic through the system, to the fun distributed complex systems problems around managing containers at scales of like 250,000 containers, to like frontend APIs, to UIs, to concourse and its automate — It's just incredible. I like to think of like how many startups are we at right now? It's just the scope of what we're doing.

You end up finding you have to organize around components and domain so that folks can go deep. It's very hard to be really, really strong UI developer who's able to pull something like PCF metrics off where you have just these beautiful visualizations of custom metrics and a learning that customers can use, and also be like a kernel hacker. These things are just very different. You need to have these vertical slices.

But then to deliver value to the platform, to deliver an end-to-end feature, that's actually a horizontal slice that can cut across a lot of these components, and that's been this really fun challenge of how do we fund and ensure that initiatives that are crosscutting actually land well and solve the problems that they're intended to solve?

[0:07:37.0] JM: I imagine landing those well can be difficult, because you're trying to integrate — The thing about Cloud Foundry and Spring is the experience is often this highly integrated experience, but from a management point of view, if you want to orchestrate teams to work on a feature together and these teams are normally going to be operating in disparate disjoint sets —

[0:08:01.7] JM: With their own sets of priorities, with their own little mini visions for their neck of the woods. How do you drive that broader crosscutting persona-based vision or theme or initiative-based vision? That's just a constant thing that we are exploring and experimenting with. We think it's actually really important, because it's sort of this meta-thing where it's easy to see Pivotal as like this creator of software or the software vendor that's delivering software that's trying to help our customers operate in a different model, but it's more than that. Our heritage, our DNA comes from the Pivotal labs consulting practice. Whereas I started in 2010 as a labs consultant and then made my way into this position over the years, but learned a lot about what it means to write code in a healthy and sustainable way. What it means to focus a backlog or a team on a product and a set of problems that deliver value.

What we promise our customers is, "Hey, if you adopt that process, if you adopt that focus on chasing value and you use this technology stack, it will enable you to move very quickly," and that's true. The challenge is doing that at scale, doing that at scale in a way where you can have independent teams that are empowered to make their impact, building their own call it microservices or pieces of the pie, whatever you want to call it. But then how do you help organize those teams into unified, coherent visions? How do you execute something that touches across multiple teams?

As we're figuring that out — I'm in conversations with customers all the time where they ask, "How are you scaling this?" The honest answer is, "You know, we're all on this journey together." It's often like we have this really good back and forths and I share things that we're doing that are working, areas where we're struggling and sort of the philosophy of how do you empower independent teams to move quickly, but then also empower folks who want to own that crosscutting thing to really have both the authority and the autonomy to be able to move and deliver value.

[0:10:04.6] JM: A lot of the people to listen to this show listen because they want to understand how these large and productive software organizations that they see, how the business actually works. So we like to mix in the notions of business along with the notions of engineering. I'd like to talk a little bit about the Pivotal business and then we can talk about how that business informs some technical decisions.

I think people are familiar with Pivotal as a company that develops Spring and Cloud Foundry and offers some services, some consulting around that. Can you like kind of break down what are the biggest revenue drivers or how does the business work and how does that guide decisions that you make?

[0:10:47.9] OF: Sure. From a revenue driver's perspective, it's primarily around helping our customers bring their workloads to Cloud Foundry, but it's not just about getting those workloads onto Cloud Foundry. It's really about — I think the way Rob puts it as we could give them a fish — So Rob, the CEO. We could give them a bunch of fish and they can enjoy their fish. They can run some of their workloads on Cloud Foundry. What we really want to do is give them fishing poles, and more to the point we want to make them people who can create their own fishing poles. We want our customers to really embrace a different way of building software that enables them to bring not just more of their workloads, but new problems, to be able to solve new problems using a method that allows them to focus on delivering customer value that leverages the technology stack that we provide to allow them to move very quickly and to explore that space very efficiently.

Now, that drives the business in the sense that it drives consumption for Cloud Foundry, and that's important. But the real deep effect that it has is it helps all these customers have these aha moments where they realize that, “Wow! There is a new way to work here. There is a new way to build software that's much less risky, because you have these feedback loops that allow you to course correct as you go,” and is just different, and a lot of the times you'll hear things like developers say, “Oh, don't take Cloud Foundry away from me, because it's enabling me to be just so empowered, that I can just solve a problem and not worry about all of the stuff that's below the value that I want to deliver in my code, and then I can learn and turn around and adjust and push my code again.” That vision of being able to develop iteratively based on reality and data is really powerful, and I think that talking about the business value as just driving consumption of the platform doesn't do that piece justice. We don't want to just have usage. We want to have usage that's really empowering our customers to just approach this whole space very differently and move more quickly, because that's what’s going to let them succeed in the long run. We want to help them be very competitive in whatever domain that they're in. Again, fundamentally, the way to be competitive is to be able to shift direction quickly in response to the changing context around us. Everything's changing faster and faster and faster, and so the only way to keep up is to be able to change along with it at pace.

[SPONSOR MESSAGE]

[0:13:25.6] 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 talk 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@softwareengineeringdaily.com.

If you're a listener 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. Send me an email at jeff@softwareengineeringdaily.com.

Thank you.

[INTERVIEW CONTINUED]

[0:14:53.0] JM: Yeah. What about the notion of being a cloud provider while also — Preparing for these shows, I kind of dove in to how BOSH works, and BOSH is the layer of infrastructure management that sits between Cloud Foundry and a cloud provider. If I'm Microsoft Azure or Google Cloud, I write ways to hook into BOSH to provide my infrastructure to Cloud Foundry. It's this common CLI that I can use to interact with Cloud Foundry if I'm a cloud provider, and I'm just interested in like sort of the business dynamics between Pivotal, because Pivotal has I think your own hosted Cloud Foundry offering, and then there are other — You provide ways in Cloud Foundry for other cloud providers to offer Cloud Foundry as a service. I'm just interested like the business dynamics of how that works, and can you — Is it situated in a way where Pivotal can end up making money off of Cloud Foundry even when a Cloud Foundry instance is hosted on Azure?

[0:16:05.1] OF: Yeah. There's a lot there to unpack. BOSH provides an abstraction that allows Cloud Foundry to run on top of basically any infrastructure as a service that can satisfy that abstraction. It's not a very complicated API. It's just a bunch of endpoints, and if you satisfy them, we will be able to drive your automation to deliver whatever it is that we've packaged on top of BOSH. BOSH is very generic. It doesn't just deploy Cloud Foundry. It deploys whatever you write to deploy on top of it. We even have customers who've started to use BOSH to deploy their own workloads in that way, but that's a side story.

So why do we do this? I think one of the main reasons is because as you see this great and exciting competitive landscape emerging of these different cloud providers and as you see companies grappling with like on-premise versus cloud and just this sense, "Gosh! It's hard to choose, and it's hard to make a big bet, and it's hard to know where we're going to end up in years from now." What Cloud Foundry brings to the table is that portability and that optimality.

If you write your applications to run on top of Cloud Foundry using Cloud Foundry services and application runtime, like you can then take what you've deployed and run it on top of any other cloud provider or run it on-premise, and it basically just transfers over. You don't have that sense of lock-in and you can start to try and explore these different things, or even better, you can have a shared interface for how you run your applications and then pick different providers based on sort of the services that they offer. Maybe you'd pick GCP because of the great machine learning stuff that they've got, but you might pick Azure because of whatever other reason. So you have that optimality and you don't have to feel like locked-in and like you have to do all of your tooling around solving for one cloud or the other, that by building on top of Cloud Foundry, which builds on top of BOSH, you have that option.

For us, again, from the business perspective, we're not charging you for the VM's or the IaaS, it's all about consumption of containers.

Regardless of where you're running, that sort of makes sense for us and it also makes sense for our partners, Azure and Microsoft, Google, GCP, because they're going to be getting revenue because you're using their virtualization, you're using their hardware. It's just mutually beneficial thing for us and for our partners.

[0:18:27.0] JM: Pivotal gets a cut even when Cloud Foundry is running on Google's infrastructure, for example.

[0:18:34.2] OF: You pay for — Yes, Cloud Foundry, to manager containers. We don't get a cut of like the IaaS spend that you have, just to be clear. That's going —

[0:18:44.0] JM: It's like a fixed subscription thing or —

[0:18:47.4] OF: Yeah, it's a subscription, basically.

[0:18:49.7] JM: Okay. I see. So if I — By the way, it's not all like, "How do you do this? What do you — Your machinations or something."

[0:19:00.7] OF: You're just trying to understand how it works. Yeah.

[0:19:02.0] JM: Absolutely. If somebody sets up Cloud Foundry on GCP, it's running on Google cloud's infrastructure and —

[0:19:11.4] OF: Correct. It's running on Google clould's infrastructure. You basically just give us your credentials to your Google cloud project and we take care completely of managing the infrastructures. We'll spin up VM's on demand. This is the real power of BOSH and Cloud Foundry. Not only do you get a platform that will run your applications and take care of all of the low value stuff that you really shouldn't be worried about. You should just worry about your code. That's the promise we make to the developer, "Here is my source code when I'm on the cloud for me. I do not care how. We just make that happen."

We also make a promise to that sort of operator persona, like the person at the enterprise that's responsible for providing this service to their developers. They have to run Cloud Foundry. They have to deploy it somehow. What BOSH does is it makes that really relatively easy. We make it relatively easy to like get Cloud Foundry bootstrapped and up and running on something like GCP. That's the day one experience. But the real value, the real power is the day two experience.

When we ship a new release, there is a high severity CVE. We'll turn it around in like a couple of days and you'll get a new release. You can, single click of a button, upgrade your cluster and we'll roll through all of the VM's. You don't have to do any of it. We'll provision new ones, shut down old ones. Do it in such a way that everything remains up and highly available. If you build your applications in this 12 factor format, like your applications will stay up. You won't see any downtime, and we'll just manage the entire IaaS for you, whether it's Google or Azure or AWS or VSphere, and it all just works. In that remark, it's sort of like making the impossible possible, like it's very hard for your typical enterprise customer to say with confidence, "We've patched everything now." But with us, to patch your Cloud Foundry is literally download the latest tile, hit apply, and everything gets patched.

[0:21:12.6] JM: So if I am Google, why don't I look at that and say — Because Cloud Foundry's open source, so why doesn't Google say, “Hey, we should get into the Pivotal or the Cloud Foundry support business.” Why don't they do that?

[0:21:27.6] OF: I think just we've been partnering, and that's been going well. We’re able to do that together, and so they focus on —

[0:21:33.6] JM: And there’s a lot of domain expertise.

[0:21:34.7] OF: Yeah, and they focus on building an amazing IaaS and we focus on building an amazing platform that runs on top of amazing IaaS. Again, it’s mutually beneficial.

[0:21:42.0] JM: Yes. Fascinating. Okay, cool. That's really clarified things a lot for me. Now that we have the business model in mind, the interface to provide at that BOSH layer to allow any cloud provider to plug in to Cloud Foundry, you mentioned that you really want this BOSH layer to be a pretty narrow interface. Why is that? Why don’t you want a more robust interface for cloud providers to really plug in many different ways their infrastructure into Cloud Foundry?

[0:22:16.8] OF: Maybe I make it sound super narrow. It's not super, duper. It's as narrow as it needs to be to satisfy the workloads that we want to run. As we've expanded the class of workloads, we've actually been adding to that API to make it more proficient and able to do more. This does have this interesting effect where we sort of take a lowest common denominator approach.

If something can run on BOSH, it can therefore run on every IaaS. So it becomes a little bit harder to differentiate between the IaaSs. With that said, there are ways to sort of pass custom metadata down. This is getting detailed. But there are ways —

[0:22:52.4] JM: That’s fine. It’s a details podcast.

[0:22:53.9] OF: Right. There are ways for folks to expose some particular configurability in their IaaS to some extent that allows them to differentiate relative to each other. A good example is all these different IaaSs have different ways of doing that. I'm going to get this wrong. That's sort of cheaper unit VM, like you can opt into like more ephemeral VM's that cost you less per hour. There're different names for these for the different providers.

You can expose that sort of configuration through BOSH, and so you can start to differentiate a little bit IaaS by IaaS. The basic functionality needs to be the same, because what we want to be able to do is make a promise around stability and upgrades. We need to keep the testing matrix as it were, as narrow as we can. Otherwise we have this proliferation of like edges that you have to cover and test.

One of the most remarkable things that we do is just the degree of testing is amazing. We have this one team called the master pipeline team. They're at the very end of the pipeline and they will take every artifact that's coming out from all of our teams, combine them and deploy them to the various [inaudible 0:23:59.8] VSphere, AWS, GCP and Azure, and they will do upgrade tests to make sure you get from A to B. They will do fresh install tests to make sure that A works, to make sure that B works. As they're doing them, it's like we're validating up time throughout the test, we're validating the everything is green.

These things are amazing and they're fully automated, and like the — What I love about Pivotal is just we're all in on testing and automation, and like we run towards the pain. If something is hard to automate, that's fun. Let's go fix that. This is where Concourse was born. It came out of this sense of while there's lots of tools out there that are really good and solid and great, like we were missing a tool that would allow us to really bring the level of automation we needed to our pipelines. It's really a project that was born out of that need here at Pivotal.

[0:24:49.9] JM: Yeah, testing Cloud Foundry on a bajillion different cloud providers, that sounds like the same problem of testing your mobile application on the 80 million android flavors.

[0:25:01.7] OF: Right. So you want to constrain yourself to something that's sand, right. Otherwise, it's too much.

[0:25:06.7] JM: Yeah. One thing I found about Spring as I was doing research for it, Spring cloud, that I just didn't know because I haven't worked with it, is that Spring cloud uses a bunch of components that were originally developed at Netflix, like these projects like Histrix, for example. What's the relationship between Netflix and Pivotal?

[0:25:27.8] OF: I think that's a good question that like Ryan Morgan can dig into a little more deeply, just honestly. But largely it's been this mutual embrace of open source, and so they've totally embraced Spring and Spring boot and we've been totally embracing the sort of opinions and pieces of infrastructure that they've been coming out with around things like Spring cloud. So that's been an area of collaboration.

[0:25:52.4] JM: It sounds pretty harmonious, because Netflix is basically — They know how to do things. They kind of see the future before it — Like five or 10 years before it happens to the rest the world. So they'd build these open source projects like Histrix and then you can just dog food them basically.

[0:26:08.5] OF: Right, and what's great is they're sort of — They make a lot of sense within the broader Spring ecosystem, and so there's a lot of consonants. These things work well together, and so we're able to bring the learnings that Netflix has arrived at within their context, which is just amazing, like the scale that they're running at and the way that they're engineering their stuff. We're able to take those ideas and then make them available to our customers.

[0:26:33.2] JM: Yeah. Now, Netflix is kind of a polyglot stack. I've interviewed some people there who are working on node stuff. Plenty people who work on Java stuff. In a modern-day organization, like Netflix, do you know how they're picking between languages?

[0:26:51.2] OF: No idea. I don't have a lot of insight into that. Sorry.

[0:26:54.4] JM: Okay. That's fair enough. One of the announcements that I found pretty interesting at the keynotes here at Spring One Platform obviously was the functions as a service work. Basically, Pivotal has a new functions as a service platform. Can you explain what was the motivation for building that and who is it for?

[0:27:16.1] OF: I think the main motivation is us really wanting to enter that space and learn, and learn with our customers to bring some of that functionality of what is it look like to have small snippets of code that can be totally managed by a platform. There're a lot of excitement around the sort of burstiness of functions as a service. I can go all the way down to zero and then all the way back up to nonzero very quickly with low latency, and what sorts of cost savings does that give me? What sort of use cases and applications does that open up?

We're super excited to really be able to introduce this, because now it puts us in a position where we can start to have these conversations with our customers and be like, "Let's explore what you would do with this. Let's explore how it can help, and let's make sure that we're both building the roadmap together so that when we do ship sort of a commercial offering built on top of Riff, which is the open source piece that we've been working on for not very long, I guess it's a relatively new project, that we deliver something that's just, again, uniquely valuable to our customers and what problems they need to solve.

[0:28:20.3] JM: You built an open source way to do functions as a service and you're working on productize. You eventually do productizing.

[0:28:27.8] OF: That's right.

[0:28:29.5] JM: Did you look at OpenWhisk when you –

[0:28:31.0] OF: Yeah. Again, I'd encourage you to talk with the team. They can speak to this much better that I can. They did a lot of reflecting and thinking about what's out there, and you can imagine there's like this big collection of pros and cons, what's good about what's out there, what do we see is lacking, and really framing the work that they're doing as like our opinion on what this looks like, what this looks like done well.

I think we'll see what sort of adoption it has. We'll see how it resonates with this space, how it plays with or doesn't with these other competing things.

[0:29:09.1] JM: Yeah. It's blue ocean.

[0:29:11.6] OF: It's exciting.

[0:29:12.6] JM: It is, definitely. I do look forward to talking to them, but I mean I don't know if you know the answer this, but the big problem that — One of the main unsolved problems that I've talked to people about in the functions as a service bases the whole cold start thing. Did you look into that at all?

[0:29:30.0] OF: It’s part of the [inaudible 0:29:30.1]. It’s like how are we going to approach that? What is that look like for us? How are we going to do caching so that the cold start is as quick as possible? I mean we have people on the team who’ve done a lot of thinking around what's the quickest way to — Say it's Java code, like, “What’s the quickest way to start a JVM? Really?” If you strip it down to its bare essentials, like how quickly can you get it up and running? That's been an area of research and trying to understand how do we take some of those learnings and like package them up transparently so you're not worrying about them. You’re just giving us a line of code and we are managing all of that.

[0:30:05.2] JM: There are some other trends. Serverless is the buzziest of the trends, but there is also obviously Kubernetes which is gaining a ton of steam. When Kubernetes came out, what was your initial reaction to it? What were the different opportunity? Because this is obviously a useful building block. How did you think about Pivotal as a company? Where were the opportunities where you could leverage it?

[0:30:33.2] OF: Sure. Kubernetes came out right as we were — We were on the cusp of shipping our own rewrite FR container orchestrator. Cloud Foundry is built on top of a container orchestrator called Diego, the name. Historical reasons for the name. Even from early on there were questions of like, “Is this is an alternative? Is this something that we should look at to sort of plug-in underneath Cloud Foundry?”

It then quickly came to the realization that the value of Cloud foundry is really around the abstractions that it provides our customers and the opinionated workflow that it has. While we could do something like that, it really doesn't add a lot of value there, and that what we've learned is to really bring these things to production at scale. It's been incredibly valuable to own the whole stack all the way down and to be able to adapt it and tailor it towards that use case so that you end up with what is a really seamless experience. We've been able to continue to layer on functionality that makes a lot of sense.

As I shared in my keynote, because of the opinionation of the platform, it does allow developers to move very quickly, but it does limit sort of the different types of workloads that you can bring. In particular, if you have something that has really complicated lifecycle requirements or networking or persistence requirements, it's not always a good fit for Cloud Foundry.

I mean we've actually found that a lot of these things, they actually do run and they run just fine. So in those cases we just bring them on. But there comes a point where there's so much complexity to run something typically because it's sort of written for a different — It’s almost a worldview. It's not sort of written to be as cloud native thing that it needs more care and feeding, that something like Kubernetes provides you with sort of these lower-level primitives to be able to manage something like that.

Really, for us, it became this is very obvious, almost no-brainer. It’s like, “Hey, our customers want to bring their workloads to Pivotal. They really enjoy working with us.” Where at the stage where we can say yes to a lot of their workloads, but to some of their workloads, there's this awkward no. Why? Let's just stop doing that. Let's enable them to bring their workloads. Let's give them Kubernetes, which is where a lot of these workloads are finding a really solid home. So let's bring that to market, but let's not just not do that. Let’s think about what the market actually needs and is lacking. When you look at it, one of the things that's really challenging is, “How do I stand up my Kubernetes? What's the right way to do that? What opinion should I have around high-availability, around how to do upgrades, around how to — How do I stay up to date? Kubernetes is moving so quickly. There's lots of goodness all the time. Every quarter they have this great release that has a lot of stuff in it. How do I keep up?”

The answer is like no one really has a very strong answer for that, especially if you want to manage it yourself. These Kubernetes as a service providers that we’re seeing pop-up, like Amazon and Google and Microsoft. They’ll be able to do that for you, but what if I want to run my Kubernetes workload on- premise? What if I want to have more control over how the control plane is configured? What have you?

Really became an opportunity for us to say, “You know, we've been in the space for a long time figuring out how to take complex, distributed systems and deploy them on-premise behind the firewall in a way that our customers can manage them and we don't have to, and that's BOSH. That's the power of BOSH again." What if we take all of those lessons and bring them to Kubernetes? What if we begin to provide that as a service and like how does multi-tenancy work in Kubernetes? Those are things we're exploring with art with our customers, and what we're were going to enable folks to do is to deploy and manage not just one Kubernetes cluster or two, but just to really turn them into pets — Sorry. Into cattle, like the opposite of pets, like dev team over there needs a Kubernetes. Just don't even file a ticket. Give me Kubernetes, enter. We spin up a cluster. Now, you have access to it. When it's time to do an upgrade to the latest version or because there's a security vulnerability and we have to patch the operating system, it's the same experience. Just download it from our distribution network and install it.

What we're doing is taking the lessons we've learned about how to package and distribute this complex distributed software and we're applying it to Kubernetes and then we're bringing that to the market so that our customers don't have to be like, "Oh, not all of my workloads will work on PCF." Now they can be like, "Oh, yeah. No, I got this." Most of them do, and they're just going to run them and then we're going to pick the right tool for the job.

[0:35:00.6] JM: And so it was a just a matter of taking Diego out, plugging —

[0:35:05.5] OF: No. That's not what we're doing. The Pivotal application service continues to run Diego. It works great at scale. We have a lot of depth of experience at running it in the way that our customers we've observed them use it. We have customers running thousands of applications and multitenant environments on a single Cloud Foundry cluster and it gives them lots of efficiencies and it's like just a small team of operators can manage that entire thing, like that's super valuable. We don't want to break that. We don't want lose that. We don't want to change that. There's no real value in changing that. It's not an or conversation in that sense. It's really an and conversation we're saying.

In addition to that, you can also spin up a Kubernetes cluster, and rather than put in front of Kubernetes an API that's more opinionated, which would begin to limit sort of the utility of this lower-level abstraction that is Kubernetes, we're just saying you can have Kubernetes, full-blown, unencumbered, vanilla Kubernetes. You can upgrade it whenever you need to. You can you can deploy multiple clusters of it and we're going to help you make that really easy.

Again, it's all about this learning journey. As we push that out, we're going to learn really quickly from our customers. What are you doing with this? How's that working for you? There's been a lot of it. We started to do some early interviews and sort of get the product into people's hands, and there's a lot of excitement around, "Oh my gosh! You mean I don't have to worry about how to manage this thing? You mean I don't have to just have one big cluster because I don't know how to deploy seven of them? Great! Sign me up."

[SPONSOR MESSAGE]

[0:36:34.4] JM: You are building a cloud-native application and you need to pick a cloud service provider. Maybe you're just starting out with a new app, but you have dreams of scaling into the next giant unicorn. Maybe your business had been using on-premise servers and you want to start moving some of your infrastructure to a secure cloud provider that you can trust. Maybe you're already in the cloud, but you want to go multi-cloud for added resilience.

IBM Cloud gives you all the tools you need to build cloud-native applications. Use IBM Cloud Container Service to easily manage the deployment of your Docker containers. For serverless applications, use IBM Cloud Functions for low-cost, event-driven scalability. If you like to work with a fully-managed platform as a service, IBM Cloud Foundry gives you a cloud operating system to control your distributed application.

IBM Cloud is built on top of open-source tools and it integrates with all the third-party services that you need to build, deploy and manage your application. To start building with AI, IoT, data and mobile services today, go to softwareengineeringdaily.com/ibm and get started with countless tutorials and SDKs. You can start building apps for free and try numerous cloud services with no time restrictions. Try it out at softwareengineeringdaily.com/ibm.

Thanks again to IBM for being a new sponsor. We really appreciate it.

[INTERVIEW CONTINUED]

[0:38:10.8] JM: This is like I am a bank and I run on Cloud Foundry and there are some new workloads that some engineers are coming up with. Maybe their machine learning workloads or some other kind of workload that for some reason doesn't run great on Cloud Foundry because of the way Diego, the container orchestration system for Cloud Foundry, because the way that works.

[0:38:35.4] OF: It's less about the container orchestration and that it's more about the constraints that we apply in order to help in order to — It's sort of we put these guardrails up and say, "Hey, if you build within these guardrails, you'll be able to go very quickly and really able to manage the software completely autonomously. You won't have to be involved." It's less about we're building something new that is a better fit for Kubernetes. What we find more and more often is we have something that's a little older or we have some commercial off-the-shelf software that's now being packaged to the Docker image, but it has all of these weird constraints, so it doesn't work well in Cloud Foundry, on PAS, Pivotal application service.

Now we can say, "Okay, we'll just bring that to Kubernetes where it might be a little more finicky to manage," but you still get a lot of the benefits of BOSH in the day two operations and you get a lot of the benefits of the speed of Kubernetes, and so it makes more sense to run it there than however you're currently running it today. Does that make sense?

[0:39:30.9] JM: Maybe. I'm a little bit unclear. So let's start with the workload. So what would be an example of the workload that I would want to run on Kubernetes as supposed to on the Cloud Foundry?

[0:39:44.8] OF: Sure. A good one that we are seeing a lot of interest in is something like elastic search. Okay. It needs persistence. It has some complex lifecycle stuff. It's generally difficult to like fully automate. Not a great fit for something that's optimized around, "Just give me your source code and I'll run it for you."

With Kubernetes, there's a lot more configurability around managing something like that, bringing it to a more cloud native world where you could spin up your container much more quickly. You can move things around more easily. I think that's been a classic example, and it's one that our customers have brought us, like, “I want to run this. I'm building applications that need elastic search. I get that it doesn't run on the platform, but what do I do? Help me.”

What we now have an answer, we can say, “Hey, here's Kubernetes. Here's the elastic search communities opinion on how to run it on Kubernetes. It’s not going to be necessarily as fully automated as what you’re used to with, with Cloud Foundry, that sort of pushbutton BOSH upgrades are. They just work. But it's deftly better than what you're doing now and it gives you that flexibility, and you can you can bring it to the platform and run alongside your other workload.” Does that help?

[0:41:03.4] JM: Yes. Now, the thing I'm a little confused about is I’m the bank, I want to set up my elastic search cluster. I want to use — I guess I want to use Pivotal as my service provider, and that's how I'm deploying elastic search. How does elastic search get deployed on Kubernetes? What is my interface with Pivotal in this transaction when I'm setting it up?

[0:41:29.7] OF: Got it. You're asking us for a Kubernetes cluster, and we’re giving it to you, and then you're turning around installing elastic search on it or maybe you’re building some automation on that. Now, what's interesting is as we have delivered, started to deliver the PKS style Kubernetes managed by BOSH, we’ve seen a lot of excitement from our ISV’s, from our partners going, “Hey, we'd love to bring our software to run on top of PKS,” and that experience will be more seamless. That will be like, “Hey, give me an instance of X,” whatever X may be, whatever partner we’re talking about. For example, GitHub is working to bring GitHub enterprise to run a top of PKF, on top of Cloud Foundry. It won't be like, “Give me a PKS cluster then I'll go over here and install GitHub,” or maybe that’s the first thing that comes out the door, but eventually what we want is to really enable this marketplace where as an operator you just say, “Hey, I want to allow folks to install service X,” and that will provide developers with an abstraction to just say, “Hey, I want an instance of service X.” Under the hood, we’re deploying a Kubernetes cluster and we are running whatever needs to be run to bring the software to up and running on top of that Kubernetes cluster. It becomes just another option within the services marketplace that we have for deploying and managing services.

[0:42:51.1] JM: Yeah, because as a bank, I don't want to be thinking about, “Okay, I want to GitHub enterprise. I want to get it on premise.” I don't want to think about am I deploying this to the underlying infrastructure of Diego or if it's the underlying structure of Kubernetes. I just want to have this —

[0:43:11.2] OF: Like, “Just give me GitHub now.” That's the ultimate goal here, and what it does is really it's less about optimality for the banks. It’s more optimality for the ISV partners, like, “Hey you can either run the top of BOSH or you can run on top of PAS or you can run a top of PKS,” but we want to do is make that a seamless experience for the bank in this in this example, right?

It's more than that. It’s not just that the bank wants to deploy it. There are services where they want to enable their teams to just pushbutton deploy something, but maybe don't want them to worry about managing it, because that's where you fall into the hole, “Hey, are you spending your cycles on something valuable or are you spending your cycles on just keeping something up and running, and do you even have that skillset?”

To the extent that we can work with our partners to automate a lot of that away so that you can reliably spend something up and trust that it works and understand how it's working and who to call if something's not working. That I think will be super valuable and it will open up — Again, it's all about velocity. Like how do we help teams at these customers focus on delivering value and learning from whatever economy or market they’re operating in in order to iterate on what they're delivering and ship a new version.

[0:44:29.7] JM: Do you think that gets you to a place eventually maybe in five or 10 years where —

[0:44:34.9] OF: Five or 10 years, yeah.

[0:44:36.6] JM: Where Pivotal thinks of itself less of — Because I think right now the Pivotal funnel is mostly you are a Spring or Cloud Foundry user and that's how you make your way into the Pivotal ecosystem, but the way they are talking out with an app marketplace, sort of maybe I don't even need to be thinking about Spring or Cloud Foundry eventually. Maybe I'm just thinking, “Okay. Yeah, Pivotal has a functions as a service platform. I like they've got some other tools that I like. I don't care about Cloud Foundry or Spring, I just want my node app that happens to run this different stuff."

[0:45:10.4] OF: Node apps run on Cloud Foundry. This is important. Cloud Foundry is polyglot. I get what you're getting at, and we're even seeing this now, some of our data services that run on the platform. There's some interest in, "Hey, I'd love to run this. I don't feel like I need all of Cloud Foundry, or maybe I have Cloud Foundry over here in this line of business, but over there we just want jump fire. We just want GPDB. Can you bring that to the service marketplace and make it just automated enough that I don't have to worry about how to manage this thing?"

We are definitely starting to see. I don't think that's 5 to 10 years away at all. I think that we're starting to see this sort of — I think Cloud Foundry will always be the emphasis at least certainly for the near term, but like the marketplace of different services, the catalog of services becoming more and more full-featured. Yeah, I think we're seeing and we will be seeing more and more people wanting to say, "Hey, I just want that one piece or this other piece, and I love the automation and I love the ease with which I can get it and enable my developers to use it." Totally.

[0:46:16.9] JM: Yeah, because we see this increase in non-commodity services, like five years ago I was thinking like, "Oh, yeah. AWS or Google or Azure, it's all the same. Just getting some compute, some storage, some blob storage. Who cares?" Today it's like really high-level stuff, and I talk to companies all the time where there like, "Yeah, I'm mostly on AWS, but I use big query, because Amazon doesn't have a big query." The natural extension of that seems to be, "there's going to be more room for more cloud providers that are offering different weirder niche, more niche service." Pretty interesting.

The other big announcement/trend that I feel is important here at this conference is reactive. I was talking to John Bloom yesterday about Spring Data, which is basically the data interface layer if you're building a Spring application so that if your querying a data source, it doesn't matter if it's MySQL or Mongo GemFire or whatever, you have a fairly consistent interface for dealing with that data. Then we got to a conversation about reactive, which is basically the idea that if there's a change to a data source that you are a consumer of, your application can respond to that change, rather than having to write some application level polling of the database, which is much more burdensome.

One of things he said that was pretty interesting was the idea that like if you want to be reactive, you generally have to build it at all layers of the stack. You have to have a database that knows how to push changes to you. You have to have middleware that know how to push the changes to you and you have to have an application level abstraction that can consume those pushed changes.

Tell me how that trend, that emphasis on reactive applications has affected software development Pivotal.

[0:48:25.5] OF: I think it's primarily been us wanting to really embrace what we're seeing emerging, and it's been emerging for a while, like this asynchronous programming model. I think the keyword in what you were talking about was consistency, like the Spring framework, Spring platform is all about really providing that consistent programming model that really well documented, easy-to-understand set of patterns that you can leverage it and get big wins quickly.

The emphasis has really been on what does it look like to do that well and to do that thoughtfully and to bring that to market so that folks, like I said in my keynote, it's not this big, expensive choice between different languages or frameworks, but it's actually something that I can wrap my head around more easily and start to leverage now because I'm used to Spring and I know how that works and I get how all of these fits in.

I think, again, it's something that I think the folks in the Spring team can speak to much more, in much more depth and detail than I would be able to. But that's really been the approach of saying, "Hey, there's a lot here that's valuable." Right now folks are feeling almost paralyzed by how do I approach this, and how do we lower that burden and that fear factor so that folks can understand that this isn't this alien thing, that it is something that you can be productive with pretty quickly.

[0:49:51.5] JM: Does that at all bring us back to the beginning of the conversation where we're talking about the fact that if you want to — Sometimes if you want to implement something across an organization, like reactive programming with this idea that touches different layers of the stack, does that require any sort of like management from on high where you have to say, "Hey, layer X of the stack, we need your middleware to become reactive."

[0:50:19.6] OF: Sure. I think there are moments in every organization where you grapple with; this must come from on high. More and more I've grown skeptical of that sort of approach. I think it's much more effective to see sort of what are the patterns emerging at the grassroots level. How do I amplify the ones that are working really well? Then at some point you get to the point where you go, "Hey. This is working and we're going to invest in it and we're going to support it and perhaps we're going to communicate that this is the future and we want everyone to move in that direction." That tends to be a more effective way to — This is just pure, like philosophy of management at this point. But I find that having that grassroots aha moment that spreads organically versus dictating top-down, thou shalt use X, especially when people get disempowered when you say thou shalt use X and they will feel obliged to use X, even though X might not be the best answer always. Then you end up with just wasted effort, low morale. So I think really like helping folks understand the values and the pros and cons of X, whatever X might be and doing that organically and then driving towards that.

I think a really good story for us was Concourse. Concourse emerged out of one of our projects. We had a couple of engineers who were just frustrated with the options that we were using at the time and we're like —

[0:51:43.3] JM: For CICD.

[0:51:44.3] OF: For CICD. I won't get into what we were using at the time. That's less important, right? It wasn't just one thing. It was like a variety of different things and we were struggling and it wasn't working, and we had tried as an organization to mandate thou shalt use one of these solutions, and people were doing it and it was okay, but there were clearly issues with it.

What was interesting was Concourse started to come out and it was, again, tailored to work the way that we want to work. Tailored to solve the complexity problems that we have, and it was really cool to see it totally emerge as this grassroots thing. One team started to adopt it, and then sort of almost broke the rules and was like, “No. This is going to be our thing now and were just going to use it.”

Another team saw that and was like, “Hey, that's really cool. Can we contribute? Can we use that?” So it spread a little bit, and at that point that created a moment for leadership. We’re like, “Okay. What do we do with this? This is signal. This is a signal to us that there is this thing emerging that we could invest in. Let's understand the pros and cons. Oh, look! The pros are pretty solid here. Folks are happier. They’re able to move more efficiently,” and so it spread a little bit more and then eventually it was like, “Hey, this is now the standard. Everything new is going to use this,” and we’d like folks to explore migrating over, and if there is a reason that you can't, that's fine, let us know and let's inform the backlog for Concourse.

We gradually got to the point where at this point just about everyone is using Concourse. We don't have much of the old stuff left. It took maybe — I don’t remember how long it took, but we got to the point where we like decommissioned what we were doing before and had pivoted over Concourse, and that was just a really good lesson for me on like how something organic can leads to this opinion forming that becomes top-down.

Now, at our scale, that's not so complicated. At larger scales, 5,000 devs, 10,000 devs, I can see how that might not be viable and how you'd need to have a more sort of top-down, “Here are the patterns that we want to follow,” but I think it's always important to like ground those in reality, right?

[0:53:52.3] JM: I worked at Amazon pretty briefly and I actually saw both of those things. So as far as innovation, it happened exactly like you just described you. You create an environment where people do feel comfortable to experiment and build stuff and they know that if they build something successful, it will get buy-in from the management. In fact, they had ways to germinate that and to nurture it. It was it was it was so built-in to Amazon's culture. You could see that this is how a lot of successful projects have started, was like somebody saw something that would be useful to make. They made it and then they got management buy-in. They got resources. Maybe they now manage their own division of the organization, but you also saw these things where they’re like, “Okay, we are going to standardize on this. We have a way of doing service-oriented architecture. We are going to service proxy this way. We're going to aggregate metrics this way. Every single team must do this." You want both of those things.

[0:54:48.2] OF: You do, and you need to the need to balance the two. You need to be thoughtful about where an opinion is in its lifecycle. Is the still in the pioneering germinating phase or are we ready to settle this landscape? Like is it time for this to become mainstream? Totally.

[0:55:03.2] JM: Yeah. I know your time is short. Last question; how is Pivotal can I change in the next five years. How is Spring in Cloud Foundry and the other associated service is going to change?

[0:55:14.9] OF: The five-year horizon is one that I find it's almost funny to really talk about this. This whole industry is moving so quickly. I think my answer is always the same and it sounds like a copout, but I don't mean it to be. Our posture is a learning posture. We're constantly learning and reevaluating. And so while we have a vision, and that vision is very clear, it's around orienting our customers towards that same posture. How do we enable learning? How to be enable fast feedback loops? How do we allow folks to focus on what's valuable and not on what's not valuable? How can we take on that burden of giving you the dialtone that you need to be productive so that you don't have to worry about getting it yourself? Those of the guiding principles that I don't see those changing. Those are fundamental to the mission. What we need to do to accomplish that? Let's find out.

[0:56:04.2] JM: Okay. Onsi Fakhouri, thanks for coming on Software Engineering Daily.

[0:56:07.5] OF: Thank you.

[END OF INTERVIEW]

[0:56:09.9] JM: Indeed Prime flips the typical model of job search and makes it easy to apply to multiple jobs and get multiple offers. Indeed Prime simplifies your job search and helps you land that ideal software engineering position from companies like Facebook, or Uber or Dropbox. Candidates get immediate exposure to top companies with just one simple application to Indeed

Prime. Companies on Prime's exclusive platform will message candidates with salary and equity upfront.

Indeed Prime is 100% free for candidates. There are no strings attached. Sign up now and help support Software Engineering Daily by going to indeed.com/sedaily. That's indeed.com/sedaily if you're looking for a job and want a simpler job search experience.

You can also put money in your pocket by referring your friends and colleagues. Refer a software engineer into the platform and get $200 when they get contacted by a company and $2,000 when they accept a job through Prime. You can learn more about this at indeed.com/prime/referral. That's indeed.com/prime/referral for the Indeed referral program.

Thanks to Indeed for being a continued sponsor of Software Engineering Daily. If I ever leave the podcasting world and need to find a job once again, Indeed Prime will be my first stop.

[END]