EPISODE 915

[INTRODUCTION]

[00:00:00] JM: Cloud Foundry is a system for managing distributed applications. Cloud Foundry was released in 2011 and has been widely adapted by enterprises that need a platform for deploying and scaling the applications that run within those enterprises. The ecosystem around Cloud Foundry includes systems for continuous delivery, pub/sub messaging and containerization.

Abby Kearns is the executive director at Cloud Foundry Foundation, a nonprofit with the goal of raising awareness and adaption of the Cloud Foundry open source project. Abby joins the show to discuss her work with Cloud Foundry and how the ecosystem has evolved with the maturity of distributed computing. We also talk about what enterprises need from application runtime platform and how Cloud Foundry has adapted Kubernetes.

[SPONSOR MESSAGE]

[00:00:56] JM: Logi Analytics is an embedded business intelligence tool that allows you to make dashboards and reports embedded in your applications. Create, deploy and constantly improve your analytic applications that engage users and drive revenue. You focus on building at the best applications for your users while Logi gets you there faster and keeps you competitive.

Logi Analytics is used by over 1,800 teams, including Verizon, Cisco, GoDaddy and J.P. Morgan Chase. Check it out by going to L-O-G-Ianalytics.com/datascience. That's logianalytics.com/datascience.

Logi can be used to maintain your brand while keeping a consistent, familiar and branded user interface so that your users don't feel like they're out of place. It's an embedded analytics tool. You can extend your application with advanced APIs, you can create custom experiences for all your users and you can deliver a platform that's tailored to meet specific customer needs, and you could do all that with Logi Analytics. Logianalytics.com/datascience to find out more.

Thank you to Logi Analytics.

[INTERVIEW]

[00:02:21] JM: Abby Kearns, welcome to Software Engineering Daily.

[00:02:23] AB: Jeff, thank you for having me. I'm so excited to be on this fantastic show.

[00:02:29] JM: I want to start with some historical context from your point of view. How has the definition of a software application platform evolved over the course of your career?

[00:02:40] AB: Well, in many ways it has evolved, and in many ways it has not. So not to sound like the old person on the show that's like get off my lawn, but I do feel like a lot of the fundamentals we're talking about today are things that I remember implementing 20 years ago. We talk about microservices a lot today as a key component of cloud native architectures, yet I remember doing SOA implementations in the early 2000s.

So I think there're a lot of things that have changed. Obviously, a heavy reliance on stateless architectures and a heavy reliance on that portability of workloads is new, but there's a lot of fundamentals that we've been using for years and years.

[00:03:22] JM: What's the vision for Cloud Foundry today?

[00:03:25] AB: The vision is to be the best platform for developers full stop.

[00:03:30] JM: What does that mean in practice?

[00:03:32] AB: In practice what that means is providing organizations a platform, and through that platform, an abstraction on the underlying infrastructure so that their development teams can write code and deploy it to production as quickly as possible. So in layman's terms, what does that mean? That means abstracting an automating away as much of the underlying capabilities as much as possible so that you can get to the point where you're deploying quickly.

So if you are, let's say, an enterprise organization that are deploying code to production every 12 months or 18 months, how do I get to where I'm deploying code to production every 15 minutes? Well, the secret ingredient is automation.

[00:04:18] JM: Describe the architecture of Cloud Foundry at a high-level.

[00:04:21] AB: Ooh! That's a tough question. It's a highly opinionated platform that's integrated a lot of the fundamentals you need to run at scale enterprise grade application. So think things like container, container orchestrator, logging, metrics, security, identity management, role-based access control. All the functionality that you need to run and manage applications in a resilient way at scale are fully integrated into the platform.

[00:04:50] JM: If I want to deploy a new application to Cloud Foundry, what are the steps that I needed to go through?

[00:04:57] AB: Well, it's one basic step which is you type in a CLI command line interface, cf-push your application. That's basically the shorthand way of saying I'm just pushing my application to this platform as a developer so that I don't need to worry. I don't need to choose which server, which VM, which container my app needs to be deployed in. The platform takes care of it for me. So from a developer standpoint, it should simply be cf-push.

[00:05:26] JM: Tell me more about the kinds of configuration of the kinds of application details that I would need to impose on my application in order to make that cf-push that one command experience possible.

[00:05:41] AB: In the most simplistic way, you just need to encapsulate your application and push it. So you don't really have to map it to a particular OS or particular – The fundamental configurations. That's all taken care of for you by buildpacks. So, essentially, what you need to do is just to be able to deploy your app, and as a developer, bind any services. So if you're deploying an application that, for example, has a dependency on my SQL, you'll also want to identity and binding to those services, and then where you're going to deploy it, what region you want to deploy your application to a cloud, etc. But at the end of the day, the goal for the platform is to make that as simple as possible.

[00:06:28] JM: So is the developer specifying those details in configuration files? Let's just say I'm a developer, I've written an application in Spring, in Java, and I want to now use Cloud Foundry to deploy it. I guess I'm wondering, what are the application details that I need to add to make it deployable to Cloud Foundry?

[00:06:51] AB: None. Come as you are, right? You'll just need to choose the buildpack with the right language, whether you're in Java, in JavaScript. Whether you're in Python, Ruby, PHP, Go, whatever. You just need to choose the buildpack. But at the end of the day, you don't have to refactor your application around the platform.

[00:07:09] JM: Are the applications that people deploy using Cloud Foundry, are they typically deployed in a container?

[00:07:15] AB: Yes. Cloud Foundry is a container-based architecture. So it is deployed in a container inside the environment. Yes.

[00:07:21] JM: What aspects of networking does Cloud Foundry abstract away or simplify?

[00:07:27] AB: Well, I think it all depends on how your environment is set up. So that's a complicated question, because networking varies depending on the environment. If you're an enterprise organization, you have a very complex networking solution, right?

But there is networking handled within the platform in terms of how applications are routed to be deployed, and there are configurations around that. Cloud Foundry leverages the work that the container networking interface has done, CNI has done. So that's integrated into the platform. But it doesn't necessarily necessitate the networking would have outside of the platform, right? So you'd still have your firewalls, if you will, or anything at the edge. So it is self-contained as a platform, but you're still having to do a lot of your normal framework outside the platform, right?

[00:08:21] JM: The open service broker is an API that allows software vendors to plug into the Cloud Foundry Ecosystem. Describe the role of the open service broker API.

[00:08:33] AB: Well, a small point of clarification. The open service broker API was actually opened up in 2016 to be used anywhere. So it is a platform agnostic API which allows you to bind any service to any platform. So it started in Cloud Foundry. We've been using the service broker for many years within Cloud Foundry, and then in 2016 we opted to open it up to be used by anyone.

So, today, every major hyperscale cloud provider has their own distribution of that broker, for example, because it's a really elegant way of binding your services. So when you think services, think things like database, messaging, anything that you're running as a service to your application. It's a good way of binding a service to your application.

This can be done – The API was really appreciated by the broader community because it's simplistic. But it also allows you to add a lot more granularity if you so choose. So maybe from a simplistic standpoint, you just want to have it bind to your application and share logging credentials, right? That could be a very simplistic way of binding a service, or perhaps it could be a way of binding that application to a service.

When you create that bind, maybe I want to make sure that that database is resilient so I am not only handling the credential management to, let's say, a MySQL database, but I can spin it up, ensure that when it's created and spun by the platform, it can be a multi-node cluster configured in a resilient way. It has limitations on who has access to it and how they can use it and how big it can scale. But you can bake all of those things into the configurations.

So it can be as simple as you would like or as complicated as you will like, but it allows you to, again, add that layer of automation and configuration management to the platform or to that API without having to do it manually, which is the way that it's handled today.

[00:10:35] JM: Is there a typical continuous delivery tool that is packaged or used with Cloud Foundry. How do Cloud Foundry users implement continuous delivery?

[00:10:46] AB: A continuous delivery is highly recommended with the platform. If you're trying to do high-velocity deployments, there's only one way to do that, and that's through continuous delivery tools, CICD pipelines. There is not one packaged in with the platform. However, you

can use whichever one you choose. There are a variety of them out there, Jenkins, Travis CI, there is Concourse. There are a variety of them out there, and the users that I see use a variety of the technologies to handle that.

[00:11:21] JM: When did you first get involved with Cloud Foundry?

[00:11:24] AB: Five years ago. When I first got involved with Cloud Foundry, I joined Pivotal when Pivotal was the original – Technically, VMware was original creator Cloud Foundry, but it was spun out as part of the Pivotal creation in 2013. I joined shortly after Pivotal was created in 2014. Then in January 2015, the Cloud Foundry Foundation was created, and Cloud Foundry was moved to an open source foundation. So a neutral body to allow for other companies besides Pivotal, like IBM and SAP, who were some of the earliest members to contribute and participate in the building of the technology and the ecosystem and the community. I joined the foundation in 2016. So I've been on the open source side for a little over three years.

[00:12:21] JM: What problems does Cloud Foundry solve for large enterprises?

[00:12:26] AB: It allows enterprises that are really trying to write more software faster. Let's be honest, most enterprises are in a way that is easy for them to do at scale. So what do I mean by that? Enterprises are really struggling to figure out how to build a new culture that is around not just software development, but high-levels of iteration and innovation at a company adapt. Platforms like Cloud Foundry allow them to not spend the time building the technology in building the platform, but leveraging an integrated and opinionated platform so that they can get their developers the ability to write and deploy code to production as quickly as possible.

Enterprises are looking to do that in a mature, scalable way that also means security requirements. So that's why I think adapting and integrating an opinionated platform, one, Cloud Foundry is key, because it allows you to focus on what's important to you, which is writing new code for your business.

[SPONSOR MESSAGE]

[00:13:38] JM: Today's episode of Software Engineering Daily is sponsored by Datadog, a monitoring platform for cloud scale infrastructure and applications. Datadog provides dashboarding, alerting application performance monitoring and log management in one tightly integrated platform so you can end-to-end visibility quickly, and integrates seamlessly with AWS so you can start monitoring EC2, RDS, ECS and all of your other AWS services in minutes. Visualize key metrics, set alerts to identify anomalies, and collaborate with your team to troubleshoot and fix issues fast. Try it yourself by starting a free 14-day trial today. Listeners of this podcast will also receive a free Datadog t-shirt. Go to softwareengineeringdaily.com/datadog to get that t-shirt. That's softwareengineeringdaily.com/datadog.

[INTERVIEW CONTINUED]

[00:14:41] JM: There has been this lineage of platform providers. I remember when I was starting to cover this space, I would see OpenStack and Cloud Foundry, and then a little bit later, Kubernetes. What was the software environment when Cloud Foundry got started? Were there other popular software deployment platforms out there? What did Cloud Foundry do differently than the other software platforms like OpenStack when it first came out?

[00:15:13] AB: Well, Cloud Foundry predates OpenStack. So Cloud Foundry was first created I think in like 2011, 2012 at VMware. So it predates all of the technologies you've named. I'd say that it was more – More of its common contemporaries would be like a Heroku or Engine Yard, right? So we're predating even the term cloud native at this point.

So Cloud Foundry, what it did indifferently was it build a lot of those technologies and integrated into a highly-opinionated platform. At the time, container solutions didn't exist beyond what you could build yourself. So Cloud Foundry had its own container technology. Cloud Foundry had its own container scheduler, its own Go router. A lot of what was Cloud Foundry was all custom built and created for Cloud Foundry and predates a lot of the technologies that we know in this space today.

Now, what Cloud Foundry has done an amazing job of is evolving and adopting these new emerging technologies as they make sense. Like on the container side, we were one of the first adapters of runC, which came out of the open container initiative, and then eventually

containerd. We've also adapted some of the other emerging technologies like Envoy, and CNI and a lot of the new Cloud Native technologies, that as they come, we're pulling into the platform. So Cloud Foundry, one, was originally custom created a lot of the technology that we talk about today, but has also evolved and adapted to the world that's changing around Cloud Foundry as well.

[00:16:55] JM: A large enterprise that might want to adapt Cloud Foundry, they've got numerous monoliths throughout the organization. They also have lots of individual services. Do they typically put all of these services on to Cloud Foundry? How does that migration of legacy services to Cloud Foundry work?

[00:17:16] AB: I mean, that's a really great question, Jeff, and that's actually part of a larger philosophical debate that I think is starting to happen. No. Cloud Foundry is not a great platform today for all workloads. It's a really great platform for microservices and cloud native applications, which is shorthand for saying small, stateless applications. But no, it's not a great home for large monolithic applications that have stateful requirements.

One of the things that I'm a big believer in is multiplatform. If you're an enterprise, you have such a broad diversity of application workloads in your environment today, and you will always will. So I am a big believer that enterprise organizations will have a multiplatform approach, just like they have a multi-cloud approach. So there's a right platform, a right technology and a right cloud for a variety of application workloads.

There is no such thing in technology as a silver bullet that solves all of my workload needs, all of my technology needs. So, as an architect or head of engineering in an enterprise organization, you're going to have to really be thoughtful about where my application workload is going and why? What do I have to work with and where am I today and where do I want to be? Really be thoughtful and intentional about the distribution of your workloads.

[00:18:38] JM: So when you're talking about those stateful monolithic workloads, we might be talking about like a legacy COBOL application, right? That's the kind of thing that wouldn't necessarily get ported to Cloud Foundry.

[00:18:50] AB: There are COBOL buildpacks. I'd say things like WeSphere, right? We're talking pure middleware at this point. I wouldn't put WebSphere or a legacy CRM application on Cloud Foundry even though there are WebSphere buildpacks. At the end of the day, think about it this way. What are you trying to achieve? We're going to go on a tangent here, Jeff, where we're going to hear Abby's two cents. I don't think every organization should have the strategy, "I'm moving everything to the cloud full stop," because that doesn't make sense. The cloud, just like some platforms, just don't make sense for every application, and you really need to be thoughtful about it. What's your goal? If you're saying my goal is to move to the cloud. Why? What is your goal? What are you hoping to achieve? Is it to write more applications faster? Well, moving to the cloud isn't necessarily going to solve for that.

Is it to minimize the overhead I have, or the real estate I have, or the footprint? Then sure, maybe that's an option. But that isn't going to be the best case scenario for every workload. So I really like to advocate a much more intentional approach to workload dispersion, because there are different solutions that are going to make more sense. I don't think every enterprise is going to move away from their large monolithic applications tomorrow, because it's not going to happen. It may happen over time or it may happen over the next 10, 15, 20 years. It depends on what you're trying to solve for.

But at the end of the day, an organization should be really introspective about what their goals are and what their outcomes and what they expect as a business from the application workloads, and then choose the platform in the cloud accordingly.

[00:20:41] JM: So if I am a larger enterprise, I've got 20 years' worth of application history. That probably means I've got some on-prem data centers, and I'm probably doing Cloud Foundry deployments to those data centers. I may also want to do some Cloud Foundry deployments to a public cloud provider. How am I choosing what workloads to deploy on-prem and what to deploy to the cloud?

[00:21:12] AB: Well, I think obviously an easy answer for anything that moves to the cloud is, is it written as a stateless application? Right? Is it written in the cloud native way? I say that's an easy one, because why are you moving to the cloud? Are you moving to the cloud because you

want to take advantage of the resiliency, the scalability, the portability of workloads within a cloud? Well, then you need to write your applications in accordance.

Moving a monolithic application to the cloud basically is saying you're moving your monolithic application from your data center to someone else's data center. It isn't going to give you any gains, unless your gain is to change your cost structure around. So when I think about what workloads make sense in a cloud platform, it really focuses on, "Okay, are you writing cloud native applications?" Then, yeah, a cloud, public or private, is a great choice for you.

Now, if you're re-factoring your applications to run in a stateless way, then yes, that can happen too. But if it's a large application, a monolithic application that you've had around for 20, 30, 40 years but still has a ton of data in it that you need and it's still working, maybe it's a mainframe. A lot of organizations are putting APIs in front of those applications and are making callbacks to them from a stateless application. So maybe that application sits in the cloud, but calls back to that more monolithic stateful application through an API.

Then as an organization, you need to look at what are the costs associated with that and what are the gains I'm hoping to make? Because, at scale, moving to the cloud is not always a cheaper solution. So you may choose for cost reasons to stay on-prem on the data center you have, or maybe you have data sovereignty requirements, or maybe you are running. Maybe you have latency requirements for an application, and so you need to run that on-prem. I think there're a variety of reasons why running your applications on-prem might make sense versus a hyperscale cloud provider.

[00:23:25] JM: Delve into more about what your perspective is for why enterprises should move workloads to the cloud. So like what I see is companies want to prepare themselves for the ability to consume cloud resources, because you can get access to AWS-managed databases, and Google-managed, BigQuery, and there're these tools that you cannot get in an on-prem environment anymore. I think a lot of the reason companies want to get familiar with the cloud is to consume those new cool software applications. But I would love to know, for a company that has plenty of on-prem infrastructure, like a bank, what would be a good reason for them to move some resources into the cloud?

[00:24:16] AB: Reasons I've heard are, yes, I have a couple of data centers. I'd like to shrink that to print down to like one data center, right? So there's a good reason right there to move some workloads to a public cloud. I'd like to take advantage of some of the regions, or like, as you pointed out, some of the natively available services and not a cloud are something I'm really excited about. Maybe it's AI or ML tooling. I think there's a lot of options around differentiating across the public cloud on the services that they provide fully integrated and fully offered through the platform.

But I think, again, it goes back to what are you hoping to achieve and why? So if your goal is to shrink your footprint to take advantage of services or leverage the broader resilient footprint from a hyperscale cloud provider, then absolutely. Moving your cloud applications to the cloud makes total sense. I just the goal shouldn't be, "I'm going to move to the cloud. Period." I think the goal is, "I'm going to move some of these workloads to the cloud, because here's the value that my business derives from that outcome."

[00:25:26] JM: If you think about the experience of a Cloud Foundry developer, how does that experience compare to somebody who is building purely on a cloud platform like AWS? What are the things that are taking care of for me as a Cloud Foundry developer and how does that compare to somebody who's working with a cloud provider like AWS?

[00:25:51] AB: Well, the short answer is you're in luck, because most enterprises that are using these hyperscale clouds are running Cloud Foundry on top of them. So Cloud Foundry, or platforms like Cloud Foundry give you that abstraction from the cloud so that you are able to take advantage of the cloud and run your applications there. But you also at the end of the day have flexibility and portability. So let's say I'm running my application on top of Cloud Foundry and AWS, which a lot of large enterprises do, and I like to maybe retarget and redeploy those applications to Azure instead. Well, I can do that and I can do it with no re-factoring of my applications and no downtime, and I can do it immediately.

So that's a strong advantage. If I am an enterprise and I don't want to be locked in to a particular cloud and I want to have that abstraction and I want to have that portability, then I might use up a platform as supposed to writing natively against the services that exist there today.

[00:26:53] JM: How has Kubernetes changed the architecture for Cloud Foundry?

[00:26:57] AB: Well, it's starting to change it now with projects like Arini, where we're working on creating a pluggable scheduler for the platform. Kubernetes is obviously been a huge phenomenon. Everyone's talking about it and everyone is super excited about the potential. I mean, I love Kubernetes. I think the technology has done a tremendous job of really elevating the conversation around why it is important to have container scheduling, container orchestration.

I do think, at the end of the day, we've gotten a little too focused on the hype and not taking a step back and looked at the bigger picture. Container orchestration is a small piece in a very large puzzle. If I am trying to figure out in an enterprise what my technology portfolio needs to look like, Kubernetes is a small piece of a very large puzzle. So, really, reflecting on this is a great piece to the puzzle, but by the way, there are so many other pieces that I also have to think about.

We as an industry tend to go from one hype to the next without really taking a step back very often. 2014, 2015, everyone, and their mother was talking about Docker. Now it's Kubernetes, starting to segue into service mesh, like Istio. So I think we're kind of jumping from one hype to the next as supposed to saying, "Hey! Guess what's happening right now? We're in the middle of rewriting the entire middleware market. All the tools and technologies that we've used for the last 20 years to get us here also currently being fundamentally rewritten and reevaluated." If I am thinking about the bigger picture, it's really what is the fundamental change that's happened in the landscape and how do all these technologies affect that?

[00:28:51] JM: Kubernetes is being put into Cloud Foundry. So there are some areas where Cloud Foundry can leverage Kubernetes as an internal technology. Can you tell me about like what advantages you can get by bundling Kubernetes into Cloud Foundry?

[00:29:10] AB: We are currently working to integrate Kubernetes into the platform. We've had our her own distribution since 2017 of Kubernetes. But we are currently looking – We're working on putting it in the platform. I will say, advantages, the challenge is that we've had a container

orchestrator and a container scheduler for many years. It's called Diego now. Before that it was called Warden and Garden.

So I think for us at Cloud Foundry, what's been the interesting balance around Kubernetes is that it's such a young technology. It's just hit its five year birthday, but it's still a young evolving technology, and we've had to really wait for it to catch up to the level of maturity, the technology we have in the platform exists today. So that's why it's taken this long for us to really start pulling it in an integrated end, because at the end of the day, we've got a ton of enterprises, a ton of governments, large banks that are running on Cloud Foundry. So, for us, we had to take the maturity of the technology as well as the capabilities that it brings to the platform into consideration.

So, for us, right now, it brings the Kubernetes enthusiasm and the quickly evolving technology that is Kubernetes because of the people that are participating in it. But at the end of the day, it's kind of taken a while for it to get to parity for the technology that we had in place today.

[00:30:50] JM: So, Dieto, the original container orchestrator, how does that compare to Kubernetes? How do they contrast with one another? If I've build a Docker container and I deploy it to Kubernetes, how does that compare with building a container on Cloud Foundry and deploying it?

[00:31:09] AB: Well, Kubernetes versus Cloud Foundry, you're comparing a container scheduler with a platform. So it's kind of apples and oranges in that respect. A more apt description would be Diego versus Kubernetes. Diego's been around a lot longer, is a lot more mature and has a lot more of the capabilities and the features that come from a mature technology of security, stability, all of those features and functionalities that have been part of Diego for many years.

So I think, at the end of the day, Kubernetes is getting to the point where it is getting closer the feature parity with Diego, but it's still a young technology. I think you're comparing a new technology with an existing technology. So there're a lot of differences in their approaches, in their architectures. But more importantly, you need to take a step back and look at the feature parity and the value that it brings to the platform.

[00:32:06] JM: Let's talk about the Cloud Foundry Foundation. Why was the Cloud Foundry Foundation originally created?

[00:32:13] AB: It was created as a neutral home for Cloud Foundry. So within an open-source software foundation, we hold the IP and the trademark. We help foster the community, the technology and the ecosystem, and we essentially – We basically allow other companies to participate in a neutral form. So that's the value in open-source software foundation is it allows people to come together in a way and build a framework of trusts for that participation.

[SPONSOR MESSAGE]

[00:32:54] JM: Looking for a job is painful, and if you were in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that is a good fit for you.

Vettery is an online hiring marketplace to connects highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies. Access is exclusive, and you can apply to find a job through Vttery by going to vettery.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you. No more of those recruiters sending you blind messages that say they are looking for a Java rock star with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job.

So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery. Vettery is changing the way people get hired and the way that people hire. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily. Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:34:44] JM: There have been many changes to the software industry since Cloud Foundry was created. As the project has adapted, some of those new technologies that have been created in the software industry, what role does the Cloud Foundry Foundation play in kind of shepherding those new technologies into Cloud Foundry?

[00:35:06] AB: We obviously pay close attention to the emerging open source cloud native technologies that are occurring. We encourage the community to participate and encourage and help build the conversations to help them evaluate, participate or adapt those technologies. But at the end of the day, an open source software foundation is essentially just a facilitator. We're not doing the development work, right?

[00:35:32] JM: Right. So what would be an example of a governance decision that would need to be made within the Cloud Foundry Foundation?

[00:35:39] AB: I like to describe what we do is more of a facilitator, the mediator. We do more like bringing people together to have conversations. But, yes, bylaws and governance are the underlying framework for how decisions are made and mandates are made, right? Governance decisions are like how are major technology architecture change is made? That's made through governance. We have what's called a PMC counsel, and that where a lot of our larger decisions are made by people nominated that run the different aspects of the PMC, the project management committees and their leaders in those areas and leaders in the community and they come together to vote on any major changes or adaption, like say incubating products or things like that.

But, yes, governance is a strong framework that outlines how we all work together. But at the end of the day, the ideas that we're all collaborating in the foundation just serves as a neutral home and someone that helps facilitate and moderate conversations to occur on the broader community level.

[00:36:52] JM: What are the primary issues that the Cloud Foundry Foundation is focused on today?

[00:36:57] AB: We focus on realistically just is the technology continuing to evolve and move forward? How is the contributor and committer community thriving and growing? The overall health of the ecosystem, is it continuing to evolve and grow? Working with our members and continuing to work with them to make sure that they continue to remain engaged and enthusiastic about the project.

[00:37:31] JM: The other foundation that I am the most familiar with is that the CNCF, the Cloud Native Computing Foundation, which manages the Kubernetes open source ecosystem. How does the CNCF compare to the Cloud Foundry Foundation?

[00:37:46] AB: CNCF is a sister project. So we're both underneath the Linux Foundation umbrella. We are structured very differently though. In the CNCF, Kubernetes is the most well-known project in the CNCF, but it has a lot of projects. Some other notable projects in there are projects like Linkderd, and NAT, and Prometheus. So there're a lot of cloud native open source projects that exists within the CNCF. So it's more of a amalgamation of a lot of different cloud native open source technologies with no strong opinion on who's the best or who the leader is. It's just a home for all these open source technologies, whereas Cloud Foundry Foundation is the home of Cloud Foundry. So we have a very myopic focus on the best ecosystem, the direction of the technology and the community that exist around that singular technology. So they are operated and ran very differently purposely so, because we're also trying to solve for very different things.

So I'd to liken the two is very symbiotic, because a lot of the emerging and maturing technologies that come out of CNCF are technologies we adapt and pull into the platform. So there is – Also, vice versa, we contributed the build packs to the CNCF last year. So there's a ton of symbiosis between the two projects, but fundamentally we operate very differently.

[00:39:14] JM: In the CNCF, you see this dynamic play out between large tech companies. These large tech companies, large cloud providers that have differing interests over what the direction of the Kubernetes ecosystem should go in. I'm not sure to what extent that kind of dynamic plays out in the Cloud Foundry Foundation. But more generally, how should the

differing interests of large corporate players who are sharing an open source project that they all have an interest in, how do you navigate the diplomacy of those kinds of relationships?

[00:39:57] AB: Well, Jeff, you've just described my job right there. That's it in a nutshell. Because that's what an open source project is. It's 100% – That is exactly what it is. It is managing that communication and mediating those communication threads so that they become drivers and propellers of a better technology.

But, yes, everyone comes to the table from every company, large or small, with their own views on what the technology should do and how it should address their unique customer needs. That is one of the challenges, because obviously everyone has different commercialization and productization ideals. But that's also one of the strengths, because what allows you to get is a diverse set of perspectives on what that technology can do, and that really fosters a more interesting conversation and a more interesting investment in the technology than you would have if it's just a single company. So it's one of those things. It's the strength and the challenge of what's happen – How an open source technology evolves, and that's what makes it so exciting too, because it really brings new ideas to the table in a diversity of thought that you wouldn't have otherwise. But you have to manage that and you have to manage what the expectations are.

One of the things that we spend a lot of time in Cloud Foundry is really reminding people there's a difference between a product and a project. The project is the open source. The core bits that we're all really rallied around and investing in the product is the thing that your company sells, and it may include that project and then some other things. But at the end of the day, there is a very distinct difference between a product and a project, and oftentimes people lose sight of what the difference is. But that's something is you're going into an open source project. You really need to reflect on what those two things are to you and how you want to invest in them and how you'd like to see them evolve.

[00:41:58] JM: How will the competitive landscape for cloud providers evolve over the next five years?

[00:42:02] AB: It's going to evolve very quickly. I mean, like I said earlier, we are fundamentally rewriting the technologies that got us here. So there is – We've all been super focused on Docker and Kubernetes the last five years. So, great. We've solved for a container and a container scheduler, but there're so many other things that we still haven't addressed yet; broader networking changes, data challenges, dev ops tooling, monitoring, management, securities.

So all of those technologies are going to change over the next five years and be rewritten we're going to see a ton of emergence of new technologies, commoditization and standardization of other technologies. I think combination of those two things are going to fundamentally change the landscape dramatically over the next five years.

[00:42:49] JM: There've been some open source projects recently that have changed their licenses or made some license alterations in response to what they perceive as AWS basically capturing too much market share of a certain open source projects total addressable market. This has turned into something of a debate. What's your perspective on the licensing debates with an open source?

[00:43:20] AB: I don't really engage in them, because I'm not a big believer and that's the debate we should be having personally. So I'll go back to the point I was just making about the difference between a project and a product, right?

There is no such thing as an open source business model, right? Open source by very definition is free. You're never going to make money on it. I think that if you are building a company that includes an open source project, either your own or another, then you should be really clear on what your business model is and the role that that open source technology plays in that business model. So that's where I've really kind of stayed away from licensing debates, because I think they are missing the point of the larger conversation, which is what is your business model and how are you competing with an open source technology as part of that business?

I think any startup that's coming into this space, particularly in a cloud native space, obviously, Amazon is going to be, AWS in particular, is going to be a competitor eventually if the

technology gets big enough or gets interesting enough or they see enough customer demand. So I think that's obviously something you have to pay attention to, but you also have to iterate and innovate pretty quickly.

If you open source part of your project, your product, you open source a technology or you're leveraging an open source technology and you're building a product around that, then you really need to be prepared to, one, define the value you're going to sell to make money on as part of that product you're selling, but also, two, be prepared innovate and iterate on top of that really quickly, because the value that you're providing is going in the customer's perception of that value is going to change aggressively really quickly. So you need to be prepared to do that.

[00:45:09] JM: All right. Well, as we begin to wrap up, in light of all these changes that are taking place in the cloud native ecosystem, in Kubernetes, in the cloud providers, how do you anticipate Cloud Foundry evolving over the next few years?

[00:45:25] AB: Cloud Foundry is going to have to evolve as well just like everything else. We are in the midst of a couple of different convergences; the digital transformation initiatives by enterprises, the evolution of hyperscale cloud providers and the emergence and commoditization of a lot of open source cognitive technologies. So Cloud Foundry by its very nature is going to have to also evolve and evolve quickly. I think it's going to grow into something bigger and more over the next few years. I'm excited to see what the community will bring to the table in terms of where the future needs to go.

[00:46:08] JM: Abby Kearns, thanks for coming on Software Engineering Daily. It's been real pleasure talking to you.

[00:46:12] AB: Same here, Jeff. Thank you for having me.

[END OF INTERVIEW]

[00:46:23] JM: SpringOne Platform is a conference to learn the latest about building scalable web applications. SpringOne Platform is organized by Pivotal, the company that has contributed to open source technologies, Spring and Cloud Foundry.

This year's conference will take place in Austin, Texas, October 7th through 10th, and you can get $200 off your pass by going to softwareengineeringdaily.com/spring and use promo code S1P200_SED. That code is in the show notes.

Attend SpringOne Platform and work hands-on with modern software. Meet other developers and software leaders and learn how to solve and find out solutions to your toughest scalability problems. Go to softwareengineeringdaily.com/spring and register for SpringOne Platform in Austin, Texas and get $200 off with promo code S1P200_SED. That's S1P200_SED, and that code is in the show notes.

Thank you to SpringOne Platform.

[END]