EPISODE 1311

[INTRODUCTION]

[00:00:00] JM: Kubernetes is an open source container orchestration system. It makes managing container clusters possible as well as deploying code changes to those containers. Microservice architecture is widely used today in large part because of Kubernetes. However, using it can require a large time commitment due to its learning curve. The company Okteto empowers developers to innovate and deliver cloud native applications faster than ever. Okteto's CLI lets developers deploy realistic replicas of their stack on Kubernetes and updates it for continuous deployments. It also manages different code environments, self-service access and container scaling automatically. In this episode, we talk with Ramiro Berreleza, founder and CEO of Okteto about managing Kubernetes clusters with Okteto.

[INTERVIEW]

[00:00:46] JM: Ramiro, welcome to the show.

[00:00:48] RB: Jeffrey, thanks. Thanks for having me. I'm very thrilled to be here as a longtime listener, first time participant.

[00:00:53] JM: Wonderful. Well, as we spoke about, the place to start this conversation is with the famous container orchestration wars. To jog people's memories or if they have not studied the container orchestration wars, this is a period of time, probably 2015 through 2017 roughly, when there were rivaling container orchestration systems and businesses built on top of those container orchestration systems. It was a messy time. It came out with the result of Kubernetes essentially being the de facto container orchestrator. Ramiro, what are your reflections on the container orchestration wars?

[00:01:40] RB: That was as you said, a super interesting time. Back then I was working for this DevOps-oriented company, and I remember we're pretty close to like the Mesos folk back then. And it was interesting because you had this kind of like – I can remember three kind of like big things. It was like CoreOS. It was Mesos. It was Docker, Docker Swarm.

[00:02:02] JM: Nomad.

[00:02:03] RB: Nomad. Yeah, you're right, nomad. There're a few of those. And then even at some point, Puppet and Chef were trying to do something around containers. It was a very interesting time. It's funny how, looking back, how we're still struggling with a lot of the same things back then. Like Kubernetes clearly won. It became the standard. Now everyone is using Kubernetes. But it's funny how the same concerns we had back then around, “Hey, how do developers use this?” Dev experience, observability, whether you wanted something to be declarative or more like program-oriented are still the same challenges that the community is debating these days. So for now, we solved some big issues like, hey, running containers at scale. Now people know how to. But it's still the how and the tooling around that is still something that we keep debating.

And I like it because I feel like we are finding more challenges as we adopt more of these technologies. And it's a fun space to be involved with. I've been around containers since around 2015 until today. And it's definitely a fun space to be at, even though Kubernetes kind of “won”. There's so much work to do there that keeps us busy every day.

[00:03:21] JM: Is the correct way to think about this like Linux won. There's not really a Linux company. There're a lot of Linux companies. Is that how we should be thinking about this where this is not like a domain where anybody's going to win. It's just a gigantic ecosystem with a lot of companies doing different things?

[00:03:44] RB: Yeah, I see the same way. I think that Kubernetes is really in a trajectory to become the new Linux. Like one of my mentors always talks about Kubernetes as like the cloud OS. Like Linux was like the VM OS or when clouds became a thing back when like AWS was starting. Linux clearly became the easiest way to run a cloud server. But now it's like Kubernetes is the way to run containers. So I think, as this kind of like ecosystem matures, we're going to see less about Kubernetes and more about companies building things on top of it. I think what VMware did for VMs I think is what we're going to see for Kubernetes. Not selling you Kubernetes, that's going to become invisible. Kind of like the same way that electricity is something that you just get from someone, but you don't really care that much. And more about what vendors are giving you value added on top of it. What vendors are solving. The typical problem it always had with policies, storage, access control, and now all these new things. Like in our case, dev experience, but many other things. And I think that's how – As this settles, that's where we're going to see a lot of interesting companies coming up.

[00:04:55] JM: Alright. So Kubernetes platform companies, I've seen a number of these. Perhaps the most successful outcome, at least the successful measurable outcome I would say is Heptio. Heptio started by Joe Beda and Craig McLuckie, who were early Kubernetes founders/contributors. Was sold to VMware for something like $400 million, I believe. And if I recall, the Heptio product was built heavily around support. And they were sort of searching for what is the software play here. They had a great support story. And they were able to sell to VMware sort of under the ages of we're just going to define VMware as container and Kubernetes strategy.

Since then, I guess Rancher is still going pretty well. I like Sheng Liang. But I look at Okteto, and it seems like you've done something a little bit novel, at least a little bit novel. What are you doing that is novel in the Kubernetes platform space?

[00:06:03] RB: The main thing we're trying to really innovate and do things differently is that we're building a platform that's very, very focused on developers. Okteto is not a platform for production. It’s not a platform where you run your services at scale. Our main focus is on how do we help developers who are building these services during the development lifecycle. Because there's a lot of tooling, a lot of platforms focused on things like software supply chain, security, progressive rollouts, all those things, some really cool stuff in that space. But when we started Okteto, there was no one building tooling for what we now call cloud native developers, which is all these teams were building microservices, who are building applications that we run on Kubernetes.

So from day one, my cofounder and I, all of us being developers, we're like, “This is a tool we've always needed. Let's go build it.” So it's very focused on all those things that developers do from the moment they get a feature assigned to the point where their PR is reviewed, ready to be merged, and then take it to production.

[00:07:17] JM: And who uses this? If I'm using Kubernetes, chances are I'm using it through AWS, or Google Cloud, or Azure. And they're going to give me lots of robust Kubernetes thingys. Why do I want to use Okteto?

[00:07:38] RB: Right. That's a great question. And the way Okteto as we see it is we see Okteto as the next layer above from those things. You go to AWS, you go to Google, **[inaudible 00:07:48]**, any of these cloud providers and you get your Kubernetes infrastructure. And that's step one. You have the compute, you have the engine. The question now is how your developers are going to consume this? And what we've seen in the market is you normally have two choices. One is your train your entire engineering team on Kubernetes. And they have to use things like minikube, or everyone has access to a cluster, whatnot. Or what we see more and more on the industry is developers kind of ignore Kubernetes, run things locally, whichever way they see fit. And then they kind of interact with Kubernetes maybe on a PR environment, typically on like a pre-production staging. So that's kind of like what we’ve seen on company. So Octeto kind of fills the gap in giving you the software you need so your developers can access Kubernetes when they're developing. And this is everything from how do you give them credentials to your clusters? How do they do things like create a namespace and deploy their application?

One of the things we kind of built early on with Okteto was this concept of a namespace as a service, which is the idea that you log in to a web app, you click a button, and that creates a namespace for you. And then you click another button, and you deploy your application. And then as you write code, you can sync your changes between your local machine, and this dev environment is running on Kubernetes. And you're constantly testing. You're already integrated. Ynd you're interacting with Kubernetes on day one. It’s something that some of the early adopters of Kubernetes kind of built on their own. Spotify has this platform called – Is it back store? Back channels? Something like that. And there's a few more like that. But most companies don't have anything like this. So they're using Okteto to accelerate their developers. To give them access to Kubernetes without having to train them on every single nook and cranny that, as you know, Kubernetes can be complex. And most developers don't need to get that deep. Most developers are happy to kind of like focus on the business logic and not become experts on containers, on networks, on policies. They just want all that automated. So that's where Okteto really helps teams ship software faster **[inaudible 00:10:04]**.

[00:10:07] JM: The layer on top – So you're trying to layer on top of essentially GKE, EKS, or ECS. EKS, what's the Azure – What's the Azure one?

[00:10:18] RB: Azure is AKS, I think.

[00:10:21] JM: AKS. Alright, whatever. Choose your Kubernetes acronym. You're a layer on top of them. You give those platforms richer, richer problem solutions. And those problems are things like credentials, secure namespaces, auto scaling and essentially a garbage collection. So the whole garbage collection idea is really interesting to me. Because there's obviously this trend in cloud cost management, right? Where everybody's spending too much on cloud. You got a startup. Your startup is really quickly. You scale up your infrastructure like it's nothing. You're serving traffic. You got a high-margin business. Who cares about the cogs? And then over time you realize your cogs have gotten way higher than they should be because you're not cleaning up your Kubernetes clusters? How do you build the garbage collector for Kubernetes?

[00:11:21] RB: That's one of the features that I like the most and kind of set it in context. The way we think about it is the other way around. Is, as a developer, you get to Okteto, and you deploy your dev environment. That's what you really care about. A copy of your application, any other services you might need, your code, all those things, right? One click, it's up and running. And when you're done, just go home. A lot of our customers, as you said, started to see is that as developers have access to more cloud infrastructure, these environments just kept running all the time. And as you do things like attaching a dev environment, a preview environment, any PR that has a copy for application for you to validate things. And as you start creating more branches, these environments start to add up. And developers, they don't like to think about these things. I never remember I have to delete this database. Like, more than once, I've been surprised with this huge cloud builds because I forgot to erase things. And every developer over does that. So what we decided to do was, "Hey, we'll give you one click experience with a dev environment. But really what we want to give you is this platform where you don't have to think about these things." You deploy by clicking a button. And when you're done, it's going to go away."

So what we did is we kind of built this this workflow engine that based on metrics, like requests, when you deployed, how long it's been running, when was the last time you accessed your dev environment. It will calculate. And after you'd reach a certain threshold, it would just run. And the first thing we do is it will scale everything down to zero. And it's one of the really cool things about building this platform on top of Kubernetes, is that Kubernetes already has all this mechanics in place, like scaling a deployment to see your replicas. It's fairly trivial. You just change the deployment manifest and it's just there. And then when the developers comes back, they can simply access Okteto. We build this integration with ingress controller. So as soon as they hit an endpoint, we kind of like rehydrate their dev environment, restore all the settings, and they're ready to go in seconds. And that was something that was really well received by our customers, because it solves, as you said, a very real problem, which is we're spending too much in cloud. But we do see the benefit of giving developers access to cloud, because some less sophisticated companies, their first reaction will be, “Well, don't give them access to cloud. They have computers. Let them use that.” But then they start to pay in production-only issues, in lack of expertise in their teams and all these things.

So as they mature that adoption of Kubernetes, they see that things like this, like automatic **[inaudible 00:13:52]** spaces after a while. Set them to see the so they consume less compute. We've enrolled an auto scaler for Kubernetes so that you can scale the nodes following more dev-oriented patterns, which don't necessarily match what you do in production. Because in production, traffic is fairly constant. You're getting head. You scale up and down. With developers, it’s mostly like you see this huge rush of deployments at nine in the morning. And then everyone goes home at six. And then no one's consuming CPU anymore. So we built a specialized garbage collector/auto scaler to deal with these patterns that we see that developers have. And they have to be automated because you cannot just say, “Start at nine. Shut down at six,” because in remote workforces people work at different times. They work in different schedules. So we had to build in some smarts, but it's definitely something that especially like large companies really like the idea of access to the developers, but with automatic cost control, rather than them having to chase people and tell them, “Hey, don't forget to stop this thing.” So yeah, that's kind of how it works.

[00:14:57] JM: So my impression of garbage collection is it's essentially an unsolvable problem. It never ends. Well, actually, I'll zoom out. When you look at the suite of things that you can – So

I'm kind of understanding what your just is now. You're basically saying, “Look, there's a lot of stuff that you want on top of your Kubernetes. Kubernetes is so far from being a cloud. So it’s like if we basically take the difference between – Okay. So it's like on one side you got like Kubernetes as it is when Amazon gives it to you. And then you have Kubernetes as you want to use it as a developer. And the gulf there is really, really big, which is why you have to write a lot of custom code on top of Kubernetes to get your infrastructure to work the way you want it to. With Okteto, you're basically saying, “Look, there is this margin between what Amazon gives you for Kubernetes and what you actually want out of Kubernetes. And we're just going to fill in as much of that margin as we can.

[00:15:57] RB: Yeah, that's a good way of saying. When we talk to a user sort of with the community about this is Kubernetes is a great container orchestrator. It gives you resources. It tasks policies. It runs your things. But when you're building software, you need a lot more than that. The namespace is a good example. You need somewhere to deploy your code. So we give you a one-click the namespace creation. But companies also needs policies, securities. So we automate those things for you. So you as developer don't care.

But then from there, Okteto bundles a lot of other things that you as a developer always need. A good example of this is when you install Okteto on your Kubernetes cluster, we install a remote build service for you based on BuildKit. And we also install a registry based on Docker’s container registry. And we put them there because we know that any developer who's building a cloud native application needs to build images, and they need to push them somewhere. So instead of you having to push things to GitHub, to AWS, and then do a round trip to your cluster, we just put them on the cluster so your deployments are way faster. And they will build all the tooling to automate all this for you. So you run one command, Okteto build, and that builds your container just as it was with Docker, but it's already in your cluster. So then when you do your deployment, it’s way faster. So we're going to fill in the gaps on all these workflows and tools that typically developers have to either assemble themselves or just don't use and give it to them.

Another good example of this is we automatically provision certificates for your endpoints, which is things that everyone needs to do, because you're testing your application with SSL. But if you have to do it by hand, it's time consuming, it’s boring, it’s not productive. Instead, Okteto does it for you. You deploy your application, and Okteto will build the containers, push into a local registry, and give you as a sell. And then you're ready to just test it, write some code, and iterate on this. And that is where we're seeing that people really like what we're building, because we give them those extra things that either they have to build themselves, which some developers like, but it's not really that productive. Like if you're a company whose building a REST API, spending a lot of time building a name space manager is not on your best interests, because that's not your business. Instead, we build it. It's our business. And then our customers use our stuff to build their own business and go faster, easier, reach out more people, all those things that they need to do to make a healthy business.

[00:18:36] JM: So your customer base is pretty interesting. You got some real infrastructure players, it looks like. So you got people from VMware, Cisco, Microsoft, Intel, IBM, Dell using Octeto. Is that right?

[00:18:54] RB: Yeah. So Okteto, at the end, we have different products. We have an open source CLI that's very focused on anybody who's building Kubernetes applications. Then we have a public cloud. That's for developers who want to build things in Kubernetes for themselves. And then we have a self-hosted version, which is called Okteto Enterprise. That is the version that you install on your own clusters. And that's what most of our customers use at a professional level. So one thing we've seen is that the kind of problems we solved around helping you get ready to do your work, that's pretty universal. Anyone who's building a microservice-based application has to do this prep work, right? You have to start your containers. You have to run your application. You have to figure out how to install your dependencies. And then when you write code, you have to figure out how do I upgrade my application? Do I use Docker? Do I use kubectl. So there’s really a big market of people building cloud native applications that really benefit from all this automation we're building.

So Okteto has our community. Everything has started with an open source project. So in our community, we have everything from developers in large corporations, to startups, to more traditional enterprises. It's really cool to see. And I think it's a testament to, as we have more developers, and as more companies build internal software, they need developer tools. And they need platforms like Okteto. And I think that's why Kubernetes has really grown as it has because, definitely, every day more and more companies need their own software teams.

[00:20:29] JM: And the kinds of experiences that these sort of lower level – LikeI think of Dell EMC, or Intel, or IBM, or Microsoft as needing things that go beyond web apps, right? Like most of the companies I talked to are sort of Airbnb, Uber, a gaming company, things that are higher level. When you're talking about the infrastructure players, it strikes me that they need something different out of a Kubernetes platform. They perhaps need something that is a little lower level. Can you tell me about the difference between those two customer types? Maybe the hard infrastructure customer type, the Cisco or the Microsoft kind of customer versus the Airbnb, or Postmates kind of customer that plays at a much higher level?

[00:21:15] RB: Right. Well, one thing to consider there is that these enterprises, they build software up and down the stack. We have people using Okteto to build cloud infrastructure that our teams in all the major clouds using some more low-level components or CLI. Some kind of like are open sources stuff. We've seen in those type of customers is that you have the infra, which what they need is tooling to help them accelerate their workflow when they're building components. They don't care too much about **[inaudible 00:21:45]** as a service, automated deployments. What they do like, one kind of like the core features of a CLI is the ability to hot reload a goal process on Kubernetes. The typical workflow where you read some code, you have to build a container, push it, redeploy your application, see the results. Instead of that, when you use our CLI, you seem to run one command, Okteto up. That synchronizes your code. It keeps it synchronized back and forth. It gives you a remote terminal into your container. So you write code, and you see the results. And this is something super useful for teams who are building something really close to Kubernetes if you're building a controller, an operator, because this allows you to test directly in Kubernetes from day one so you don't have to spend time building mocks, or trying to replicate what a cluster of looks like in your local machine. You're just developing directly in a cluster. And actually that's how we started the company with this in mind.

As we talk to our customers, as we see the market of Kubernetes evolve, then you have, as you mentioned, all these, the Airbnbs, the Stripes, all these companies who they're building high-level components. And what these companies value, especially as they scale, is making their developer teams more efficient. The Stripe founders have mentioned a lot of times that they see their developers as like every developer contributes to the GDP of Stripe. And they know that investing in tools that makes developers go faster, it's good for the bottom line. So that's where we see a lot of the adoption of especially our self-hosted version of Okteto, which are companies who they want their teams to go faster. They want their teams to not spend time trying to figure out how to do kubectl apply, where to get their manifest from, having to learn Kubernetes. If you're a frontend developer, learning Kubernetes is not productive. But if you have something like Okteto, you can just log in, click one button, deploy your API and then you can directly work against that API. And then you're being more productive. You're taking advantage of Kubernetes even though you're not using it directly. And that is one of the kind of core missions for us in Okteto is enable all these developers to take advantage of all these cloud native technologies without them having to be experts on every single thing, which is a very common topic as technology matures, as it gets adopted, it has to look like it has to be transparent, right? Like you have to benefit from Kubernetes without you even having cubectl installed. That is, for me, the gold standard of any technology. And I think PMs got to that point what VMware was doing. Docker did the same thing for containers, where you just run Docker up, and that gives you a container. And we hope to do the same thing for Kubernetes as a dev tool.

[00:24:36] JM: Can I ask, what was the ideation process for this? Like how did you stumble upon this? There's a very like specific layer of abstraction that you're working in. So I'd love to know how you got there.

[00:24:51] RB: It's funny, because when I look back at my career, it's funny because this is a problem that I've been facing for a very long time. I started my career at Microsoft. I was actually part of the team that was building Azure. And back then, one of the biggest pain points of my team was the service bus, was the lack of a realistic dev environment. Like we were building these things for Azure, but developers had to write all this like .net code locally and then spend some time waiting for it to get deployed, and then run your tests. Then you found something that broke, and then kind of like start over again. I saw that in Azure.

Then I moved to these startups. I kept seeing the same problem. And then right before Okteto, I was working for Atlassian. I was part of the team who was building HipChat. And we saw the same thing. HipChat was doing well at some point. We're hiding a lot of engineers. And we start to run into this problem where you're hiring engineers, your team is growing, but your tooling and your engineering practices are not growing. And you start to hit all these problems around quality, around dev velocity.

So my experience with that, and then at the same time, my two cofounders, Paolo and Ramon, Paolo was at Docker at the time. Ramon was at Google at the time. When we chatted, we kind of kept hearing the same problems that I had in my teams was the same problems Ramon had on his team. He was part of Gmail. And Pablo was part of what was Docker Cloud at that point. So we kind of keep hearing everyone has the same problems around, "How do I get started? How do I get a realistic dev environment?" How do I run my tests?" So all of that kind of like gave us the initial idea for like, "Hey, this is a problem everybody has." And in those days, I'm talking maybe three years ago. We were really deep in kind of like containers and Kubernetes. So we said, "Hey, this is a problem that everybody. This is a problem that, especially, as you move to Kubernetes, it becomes harder, because now you're going from everything runs on my machine to everything needs to run on this orchestration software. It was a spread around tens of hundreds of nodes." So we saw that's the opportunity to say, "Hey, all these problems we're seen are going to get worse as people adopt Kubernetes. Let's do something about it." So we quit our jobs. Started building this open source project. We got some really promising early traction. And through the community, through talking to prospective customers, and through our previous experience, we ended up building what now became Okteto. As you can imagine, we started with something fairly different. At the beginning, we were building a more traditional platform, something closer to Heroku but for Kubernetes. But slowly, as we got feedback, as we iterate in our ideas, we ended up with this idea of like, "Hey, what we really need is not an end to end platform. It's a platform for developers for the development lifecycle." And that's where we are today.

[00:27:49] JM: Alright, you got my attention with Heroku for Kubernetes. I love Heroku. I just love it. I love the platform. It's an undying love. I've used Heroku competitors. I love those as well. But Heroku holds a special place in my heart as basically the first place that I deployed an infrastructure to without going completely insane. I think before that I tried to use Amazon Elastic Beanstalk one time and just went completely – Like I spent all day trying to use Amazon Elastic Beanstalk. And then I tried Heroku. And I was just very – It's an experience of love at first sight. I think the Heroku for X analogy will never go away. It doesn't matter how many Firebases there are. It doesn't matter how many high-level things there are. It's always going to be Heroku.

Honestly, it's almost like an Apple-level brand. I think Salesforce made one of the most legendary acquisitions of all time. I mean, who knows? How much do you think Heroku makes for Salesforce? Like it's so hard to know, right?

[00:28:44] RB: It's hard to know, because – And I agree with you 100%. I remember there was this thread on Twitter the other day. Someone asked what's kind of the first time you felt like magic using a tool? A lot of people, myself included, were like, "Yeah, Heroku Push was like wow!" You have a repo, Heroku Push. It just runs. I don't know. I don't know if it was a good or a bad acquisition, to be honest. That's a topic that I've spent a lot of time discussing –

[00:29:10] JM: What? How is that up for debate at this point? Are you kidding me?

[00:29:13] RB: Because I feel like Heroku could have done a lot more as an independent company.

[00:29:17] JM: No. No. No. Okay. So they shouldn't have sold. I wish they wouldn't have sold, of course.
It would be amazing. Okay.

[00:29:21] RB: This is what I mean. Like I think that Heroku on its own could – I mean, who knows? It could have been a huge thing. I think the Salesforce purchase –

[00:29:29] JM: It could have been as big as Apple. It could have been as big as Apple. It could have been as big as Microsoft. 100%. 100%.

[00:29:35] RB: I think so. But at the same time I feel like I know some people that work with Salesforce and the plans they have for Heroku. I think it's interesting. People get into –

[00:29:45] JM: I completely agree. I completely agree. I haven't counted Heroku out in the slightest.

[00:29:49] RB: Oh, no. No. And honestly, like I think from a developer experience perspective, like 10 years into this, I still see them as the golden standard.

[00:29:59] JM: I so agree with you. It's kind of – Gosh. Oh, man, we could go so deep on this. Yeah. I mean, continue, continue. I want to hear your thoughts a little bit deeper on this.

[00:30:10] RB: No. I feel like they really nailed kind of like the shape of CLI, the way the commands just flow. For me, what's really valuable is that it really gets you to work really fast. It's like deploy your application. It's no longer a big deal. As you said, you don't have to get a VM. You don't have to get EC2. I think, for me, Heroku was an evolution in what serverless is trying to do, which is the idea. And I think that's where eventually we'll get to, which is, "I have code, run it." I don't care how. I don't care where. Just run it. Give an endpoint. Let me use my application.

And Heroku really did – I remember when I was in HipChat, we were using Heroku to build a lot of our chat bots, because it was so simple. It was like, "Oh, **[inaudible 00:31:01].** Sure. Add one thing to the manifest." "Oh, I need logs. Add another thing." And the way they build the marketplace was it was super easy to onboard all their services. Back then, I remember we're using Keen for analytics. And everything just flowed so nicely. It was it was very well-thought, especially for small dev teams. I think the problem with Heroku was it probably didn't scale well for enterprises where you have to get into procurement, you have to get into approvals for everything, compliance, whatnot. But for small teams who don't have to go through those workflows, I feel like that way of deploying software was great.

Actually one of our investors **[inaudible 00:31:38],** he was part of the Heroku team. And when we talked about it, it's super cool to hear what they have to say when they were building all these stuff. It feels like they were ahead of their time, right? I feel like most of us are just catching up to all these concepts around dev ergonomics that they had many years ago.

[00:32:00] JM: And they have retained. I think it's very impressive how they've managed that acquisition. I've done enough interviews with people from Heroku to know that that product is so hard to build at scale. It's hard enough to build basically in any context. But at scale, it's just insane. And have you used Firebase? Did you ever use Firebase?

[00:32:19] RB: I have. Yes, I have. We use Firebase internally for a few things.

[00:32:22] JM: Oh wow! Okay. Tell me what you use Firebase for.

[00:32:26] RB: So in our cloud, we use Firebase for some of our internal systems. We use it mostly the store part of it. It's just one of the things that I really like. When we built Okteto, one of the first things we decided on was no database, Kubernetes sort of database. It's user database. So everything, if you to okteto.com, everything is in a database. It's in Kubernetes.

But at some point, we do need it to do, "Hey, we need to store certain things outside of Kubernetes, logging information, analytics, secrets, configuration values, all those things." I'm looking around and I was like, "Okay, what gives us –" Kind of going back to our core topic of how can we be more efficient? And we were like, "Yeah, running a database is not efficient for us, because that's not our business." So we ended up one of our engineers **[inaudible 00:33:11]** ended up saying, "Hey, Firebase is a really good fit for this. We just put the values there. It scales. Put value in. Read it back. And it's so simple." And it just works well. It works well on scale. And for me, it's kind of like the same way we decided to run our stuff on something like GKE versus rolled your own Kubernetes clusters. Firebase just made sense.

And I really like all those tools that just give you endpoints to do things. Twilio and Stripe are, of course, the better known examples of this of, "Hey, give an API to replace the entire part of my business." But I do like Firebase. Supabase was trying to do the same thing. But open source are great ways to just make developers more efficient and in a way that's easy to use. I think Firebase is not as easy to use as other things. But once it's set up, no issues. I shouldn't say anything. But so far, who knows?

[00:34:08] JM: Here's my take. I think that Heroku versus Firebase is one of these classic X versus Y things that we can always point to to try to express an analogy. It's like a Microsoft versus Apple, an open versus close, iPhone versus Android. There's something different between the two. I don't know exactly what it is. I think the difference is they're both sort of a backend as a service thing. But the Firebase is like we're backend as a service. And our first-class citizen the you're building everything around is our crazy do everything database. And then Heroku just says, "You're deploying servers. And we help you deploy servers."

[00:34:49] RB: I agree. I think, going back to those days, it's funny how you can see that the initial targets they had, like their target user, really molded how those things work. Because back in the day, Heroku was very focused on like Ruby apps, the whole 12 factor app. It was you have a monolith. I remember in the first early days of Heroku, even deploying a multiservice application was not trivial, because it was very clearly built for you have a Ruby, you have a Rails app, run it. So a lot of their logic came from there. And I think, at the same time, Firebase was very focused back then on like, “Hey, you're a mobile developer. You're building a mobile app. You need this database that does everything.” And it just worked, because, yeah, back then when we had sophisticated mobile apps, a lot of the logic was, “Hey, I have data. I need to store somewhere. And I need to retrieve it later with a key.” And I feel like even though they seem similar, like they're both backend as a service kind of thing, I feel like as you start to use them, you start to see how a lot of their dev experience is very much influenced by these early use cases. I think, today, if you go to Netlify, it’s is the same thing. Netlify, I think, is very focused on frontend, static pages. And everything they do makes that easier. I think that's where this platform’s win is by taking this very specific use cases and just been better than anybody else at it. At least that’s kind of my take on Heroku versus Firebase.

[00:36:19] JM: Well, now that we've teed up this conversation, so you are trying to do Heroku for Kubernetes. It's an admirable goal thinking about the Kubernetes platform things. Do you take that approach? I mean, the closest thing really seems to be Rancher. I think Rancher – I think the biggest problem with Rancher is, to my mind, and I like Rancher. I really like Rancher as a company. But my biggest problem with Rancher is I think they had to replatform. I think they were originally on their own thing, and then they replatformed to Kubernetes. And that to me is a very difficult challenge. I would assume that they're still trying to get that to work.

[00:36:56] RB: Yeah, that's true. They have their own platform, right?

[00:36:58] JM: I think they did. I think they did.

[00:36:59] RB: At some point. Their own orchestrator I think.

[00:37:00] JM: Yeah, because they were around during this time. And then they saw the writing on the wall and they pivoted to Kubernetes.

[00:37:06] RB: I've seen a few tries. I think Rio was what Rancher was trying to do. It was really good.

[00:37:11] JM: Oh! What was the other one? There was the other one that Microsoft acquired.

[00:37:15] RB: That was Deis, Dapper?

[00:37:17] JM: Deis. Yeah, Deis. But that also was not Kubernetes. That was just a Heroku-like experience. There was also – Anyway, yeah, go ahead. Go ahead. Railyard. Railyard was – Well, no. Railyard was not – Anyways, sorry. Go ahead. Go ahead.

[00:37:31] RB: There's many of them. But I think it's still a problem. Like one of my cofounders, Pablo, was working in this company called Tutum. They acquired by Docker in kind of like the container wars. And Tutum was kind of the same thing. It's like, "Hey, I have a container. Run it." Don't make me go through infra choices. Just run it. I think it's still a goal. I think the challenge with Kubernetes, and one of the main reasons why we decided to not get into the Heroku-like thing and focus on what we're doing, is that Kubernetes is a lot broader than other platforms. So requiring you to have an application that looks very specifically doesn't work well with Kubernetes.

Because with Heroku, it was, "Hey, your app has to look this way. Has been a repo, you need to have this structure. And it just works." And I think that worked really well to the Rails community, because they were used to this kind of configuration enforcement kind of thing. Kubernetes people are running everything, right? You have people building like this sophisticated ML workflows, batch jobs, containers, API's, everything. And you have Helm, you have Manifest, customize all these things. So I think anyone who's trying to build a platform right now, and I think **[inaudible 00:38:42]** is doing a really good job at this. It's a big challenge, because you have to push both the platform and the application format. Maybe in five years, if we all as an industry agree on an application format, it might be easier. And that's why for us it was very important as we were building Okteto to be build Okteto in a way that you can bring in any application and use it. So Okteto supports everything, Helm kubectl, Docker Compose, single containers. We don't care about the format of your application. You can bring everything and you'll still benefit from all of this workflows I was talking early on. And I think that's what really sets us apart from kind of like the Heroku-like of the work, which I hope someone names it, because I think it's needed. Every time I go back to writing Kubernetes manifests, I'm like, “Why are we doing this? It’s so complicated.” I do not see the rest of the world doing this every day. So hopefully someone will come with a format. But until then, I think platforms like Okteto who are more liberal in what they accept have a much better chance of success than more forcing you in a shape kind of platforms.

[00:39:52] JM: Again, if I'm looking at the productivity features – Actually let me ask you this. What do you I think of the term GitOps?

[00:40:03] RB: I like it. I'm a big fan of doing things through Git. It's funny that it took this long to kind of get this brother adoption, because when I was talking to – Every time I talk about GitOps, I was like, “Yeah, this is not new. It's been going on for a very long time.” Kind of going back to Heroku. And there were other kind of Git-based deployments back in the day. So it's interesting that is now becoming such a big thing. But I think one of the main reasons for this emergence of GitOps is because Kubernetes manifests are so hard to manage that Git makes it a lot easier. Like having your Helm release file on a Git repo. And every time you update that, it just triggers a deployment. Gives you this very nice separation between development and production. It gives you an inbuilt audit log, because you can see Git’s history.

I think it's definitely a step forward in DevOps and in all this kind of journey with **[inaudible 00:40:57]**, with infrastructure as code and all that. And I feel like anything that makes it more accessible to everyone, it's great. Because I think one of the downsides of this DevOps kind of revolution we saw in the last 10 years or so was that it was still very much gated to anyone who had an operations background, right? Like using Chef or Puppet. It was not for everyone. Then Docker came, and now all these things with GitOps. It makes it more accessible. Commit, and it gets deployed. That's something that any developer who uses GitHub understands. And I like that a lot.

Again, going back to the topic of more and more developers are joining the workforce every month, every year, the more accessible all of our tools are. The more everyone can be productive. And the more they can build cool stuff. I don't want people to waste their time trying to figure out a YAML file. I want them to build cool software and solve cool problems for me.

[00:41:55] JM: How do you go from this place that we talked about a little bit earlier, where I see you basically as providing a lot of extra sauce on top of these hosted Kubernetes platforms on name your cloud provider? How do you go from that to the Heroku of Kubernetes? Or are you already the Heroku of Kubernetes?

[00:42:19] RB: I would like to say we are, but we're not there yet. I think for us, the path forward is really making sure that what we’re building enables developers. What's very important to me, and one of the things that I can internally track is we want to make it easier for developers who have no experience with operations or Kubernetes build cloud native apps. I think that is the path forward, is kind of taking the spirit of Heroku, which is we're going to give you this tools that are very well integrated, that are very easy to understand, very easy to use, very useable. But at the end, they are the tools where the success is going to be if developers are using those tools to build software, I think that's always the dilemma with Dev tools, which is Dev tools are a means to away, right? You want people to use your Dev tools not because of the Dev tools, but because of what they can accomplish with those Dev tools. So that for me is going to be – If we're ever kind of can claim, “Hey, we're the Heroku of Kubernetes,” it would be because of we have all these users. We're building amazing stuff on top of Kubernetes. And they don't even care it’s in Kubernetes. They say, “Hey, I have these great tools. They enable me to go fast, to iterate, to be productive, and to really be able to express my thoughts in software.” For me, that is really the path forward. And today we're very focused on kind of like building these blocks and giving you all this one-click deployment, giving them environments, automating as much as we can. And as our platform matures, we’ll be getting more into other parts of life cycle development, of the software life cycle development cycle, like how you run your tests? How do you install your tool? How do you interact with your IDE and your remote Dev environment and all this other stuff?

[00:44:05] JM: How do you get people using this? Is that hard?

[00:44:09] RB: Yes or no? It was hard at the beginning. We were part of the kind of Docker Kubernetes community. So our early days, we’re very focused on talking to the community and kind of like trying to explain why the problems we were solving were worth solving and why it was good for them to adopt those tools. Now, as more and more people kind of started building microservices, containers Kubernetes, that pitch becomes easier. I think, right now, the biggest challenge for companies like ours is that there are a lot of Dev tools companies building things for Kubernetes right now. So from a developer perspective, it can be hard to tell which one you should be using. And every tool is good for something and not so good for other things. So definitely the challenge for us is how do we explain to our users and to our community, "Hey, we are really good for this five things. If you have this kind of problems, you should use Okteto. It would make your life easier." And that is a challenge.

I think condensing – I was talking to somebody earlier today, and we're talking about how hard it is for Dev tool companies to really condense what you do to like a landing page or like a single paragraph. But it's a challenge we have. But it's something that we're – As always, talking to your community, talking to more developers is the way to go. And that's what I love about – It's got a new class of companies that are building communities, that are building on open source. It just changes the conversation compared to the old school sales-driven approach. So that makes it easier.

[00:45:39] JM: What is the biggest problem in the business today? Is it expanding sales? Is it going deeper with your existing customers? Is it engineering challenges? Is it marketing challenges? What's the most difficult part of your job today?

[00:45:56] RB: I think there's kind of a twofold challenge. One is, from an engineering perspective, one of the biggest challenges as you build a platform like ours is that we have all these ideas, and we could build 20 different features. But knowing which features to build and how they fit together is a big challenge. You could have a CLI with 35 commands. But if they don't make sense within them, or a dashboard that has all these panes that didn't have anything with each other, you're doing a disservice to your users.

So one big challenge for us is how we enable all these scenarios while keeping a user will experience. And the other big challenge today is with marketing. It's how do we get people to understand the value of Okteto by visiting our website? Because once we have developers using our tools, they get the value. They run through our kind of samples. They try with their own apps. They see the benefits. Well, the challenge is, when you have a blog post, when you have – You go to okteto.com, I only have five seconds to convince you to try it out. So how do I do that? And that's a super interesting challenge. It's something that I come from an engineering background. So I've never focused on this part of the business until now. And we have a great content designer in our team that makes all this so much better. But it's still a challenge. And it's something that I feel like will continue to be a challenge as the company matures.

[00:47:27] JM: Let's take a step back. You have been in the business for a while. You got a great resume, a lot of interesting stuff. So I just want to get your perspective, your unique perspective on when you look at the world of backend infrastructure today, what is exciting to you? And what do you see around the corner?

[00:47:48] RB: One of the things that I'm super excited about is WebAssembly. Like the whole Wasm thing, competing at edge. And I'm still a big believer in serverless. I think the querying version of serverless is still kind of like not where it should be. But the idea that you can now run these very complex computations in places that are like super close to where I'm running, where I need to consume them in a way that's easy to manage and easy to support. It's super interesting. I think the stuff that Cloudflare is doing, where you can run business logic in a secure way on the edge of the CDN. It has the potential to change a lot of how we see computing, because now you're moving from everything, from this huge data center next to a waterfall, to, “Hey, I'm going to split my application and all these super small pieces spread all over the world. And that's where my users are going to consume it.” And it's super interesting from an architecture perspective, from an operations perspective. Just imagine how you do a rollout when you have functions in, I don't know, 100, 200 endpoints all over the world? How do you ensure that versions are compatible? It's a super interesting challenge. And I think it's also a very real problem. So that's something that I'm super interested to see how it evolves. And the idea of serverless of, hey, going back to, “I'm writing code. I want it to run efficiently, fast, without me having to do anything.” That for me is a very interesting place to be at. I think the work that OpenFaaS is doing, Knative, Lambda, it's definitely going in the right direction. And I can't wait to see what the next generation of those tools look like.

[00:49:31] JM: Awesome. Okay, now the last five minutes, I get to ask you this stupid question. I was at a Kubernetes conference three or four years ago when I realized something. The

Kubernetes people and the blockchain people don't talk to each other. Why don't they talk to each other? They don't seem to even have respect for each other? They might even have disdain for each other. What's going on there?

[00:49:54] RB: That's a really good question. I never thought about it. But you are completely right. I don't know. I feel like –

[00:49:59] JM: It's just so funny to me. It just feels like the blockchain people think that the Kubernetes people are squares. And the Kubernetes people think the blockchain people are just crazy losers. I think that's what's going on.

[00:50:11] RB: think there is something to it. I feel like Kubernetes became like enterprise so fast. I remember same thing going to KubeCon. And I went to the first two KubeCons and it felt like this indie conference.

[00:50:26] JM: Oh my God! It was so good. It was so good, man.

[00:50:27] RB: No one knew what is Kubernetes. It was good. I know, it was really cool. And then from one KubeCon, and I think it was maybe Austin to Seattle. It was like, “Boom!” Every major vendor is sponsoring. You have this huge hall like every well-known brand trying to do things that didn't even make sense, but they were doing it with Kubernetes. And I was like, “Wow!” You just cross this chasm super-fast. So I think that the blockchain people still see themselves kind of almost like this – Going back to the Apple analogy, kind of the pirates versus the Navy. And I think they see Kubernetes as a very enterprise as toughy thing. And most people who are not involved with blockchain I think see blockchain as this crazy Wild West of things with no use. But I honestly feel like they could benefit from each other, because Kubernetes gives you compute, and it gives you distributed compute, which is what you need for a blockchain. So it feels like almost like a match for each other. But you're right, they don't like each other for some reason.

[00:51:29] JM: Anyway, Ramiro, this has been a really interesting conversation. Anything else you want to add? Anything else you want to talk about?

[00:51:34] RB: No. It’s been great. Just wanted everyone who's listening to invite them to try out Okteto. We’re a developer tools company built by developers.

[00:51:42] JM: What’s the sweet spot? What's the sweet spot? Like what's the biggest pain point that you hear when people say, “Oh, Okteto saved my life because of X.” What do they say?

[00:51:51] RB: Right now, and this sounds kind of funny, but actually we have a lot of customers doing this, is if you have a Docker Compose that's too big to run locally, five, six services, a couple of databases, give Okteto a shot. Because just moving your Dev environment from running Docker compose locally to run the same thing, but on our platform, makes you go faster, makes your battery last as twice as much. It's a really cool use case. So if you're struggling with Docker Compose locally, if your machine kind of slows down to like snail's pace. When you run Docker Compose up, try Okteto, and you'll see some really quick benefits.

[00:52:28] JM: Well, Ramiro, thank you so much for coming on. It's been a real pleasure talking to you.

[00:52:32] RB: Thanks to you. It was great. Fun conversation. Love talking about the container wars, Heroku, all this good stuff. Thanks for having me. I’m looking forward to –

[00:52:39] JM: is there ever going to be another container wars? Are we ever going to have that again? I really hope not, right?

[00:52:44] RB: Oh, I'm sure we'll have whatever the next container looks like. I'm already –

[00:52:49] JM: What is that battle? Is that –

[00:52:50] RB: WebAssembly runtime.

[00:52:51] JM: WebAssembly. I was thinking of WebAssembly. I was thinking of WebAssembly. Maybe something in data engineering news. Maybe it’s the Airflow orchestrator. Have you looked at Airflow? What’s going on there?

[00:52:57] RB: Maybe. Yeah. Yeah. Yeah. That could be another one. There're a lot of really cool things going on data orchestration. I mean, we already had –

[00:53:02] JM: That space is such a mess.

[00:53:04] RB: I know. We had container wars. We had VM wars. We had Java runtime wars at some point. So I'm sure there'll be something else where we engineers love to debate each other and love to pick winners. So I'm sure there'll be something.

[00:53:15] JM: I think the data orchestrators are going to go to war. I think Dagter and – Not to amp it up. I'm going to amp up the cage match, because the cage match is going to be aired on Software Daily.

[00:53:28] RB: Yes.

[00:53:29] JM: It's going to be bloody.

[00:53:30] RB: Dagter versus Airflow.

[00:53:31] JM: Dagster versus Prefect. Airflow is like a passive observer. I mean, Airflow is just like, "Yeah, I'm winning. I've already won the game."

[00:53:42] RB: That'll be interesting. I'll be looking forward to hearing that episode.

[00:53:44] JM: Because I think Airflow has reached – Because Airflow is so proliferate. It's like businesses around Airflow are never going to go away. I don't think. Anyway, I mean, it's just seems so – Anyway. Okay. Well, great to talk to you. Have a great day, Ramiro. Talk to you soon.

[00:53:58] RB: It was great meeting you. And thanks for having me here. It was fun.

[END]