EPISODE 1212

[INTRODUCTION]

[00:00:00] JM: Cloud platforms are often categorized as providing either infrastructure as a service or platform as a service. On one side of the spectrum are infrastructure as a service giant such as AWS which provide a broad range of services for building infrastructure. On the other range are PaaS providers such as Heroku and Netlify which abstract away the lower level choices and focus on developer experience. DigitalOcean has carved out a sizable niche in the cloud hosting space by targeting the middle ground, a streamlined cloud platform built for developers which still offers the ability to choose, customize, and manage infrastructure. The release of DigitalOcean's app platform takes this goal a step further. The app platform allows users to build and deploy an app or static site directly from GitHub directly onto a DigitalOcean-managed Kubernetes cluster. Teams can access the power, scale and flexibility of Kubernetes without having to worry about the complexity of managing a cluster themselves. The app platform gives developers the choice of how much of their infrastructure they want to control and how much they want to be provided by the platform.

Cody Baker and Apurva Joshi work at DigitalOcean. They join the show today to talk about why DigitalOcean stands out in a competitive cloud hosting space and what the value proposition for developers is if they're interested in the app platform as well as the overall platform as a service industry.

[INTERVIEW]

[00:01:19] JM: Guys, welcome to the show.

[00:01:21] AJ: Hey, thank you for having us.

[00:01:22] CB: Thank you. Nice to be here.

[00:01:24] JM: So DigitalOcean has been around for a while and it's largely been used for raw server infrastructure. But over time you've developed a higher level solution, and I think this kind of higher level layer to cloud solution has been implemented in the past and in examples like Heroku, Netlify, these companies that have built a layer 2 cloud providers where they emphasize a different developer experience on top of raw server infrastructure. Can you tell me your thesis for building an app platform on top of your raw server infrastructure? What did you want to deliver to the user?

[00:02:11] AJ: Right. That's a great question, right? So if you think about how the cloud business is evolving over the last decade or whatnot, right? As people try and invest more time and energy in writing beautiful code and doing things that are more important versus managing the infrastructure, that trend has been evolving and is catching up, right? So it's just not Heroku or Netlify. They're coming up with the creative ways of solving a very specific scenarios or the web hosting per se, right? Netlify is going after J the fight is going after Jamstack. Heroku is more open. They started with Ruby. Speaking of my past life, I spent 17+ year at Microsoft and I was fortunate to be part of the team with Azure and the one behind building Azure app services which is a PaaS kind of a platform that abstracts the notion of managing VMs and providing that experience back to your end user where they can focus on just writing code and all the orchestration pieces are taken care of for you automatically. It's meaningful, right?

And if you look at any cloud business as a whole, majority of the customer segment that are hosting and the applications that more than 50 or 60 percent, they're actually web apps or the APIs or whatnot, right? So the drives are really high in a large target market for that kind of an application. So when we were thinking about serving our customer base and as the new devs and the individual devs and startups are building new applications, they take in cloud native as the first a bet. We wanted to build a PaaS platform that is unique and different and is actually built using all the different cloud native technologies versus what is out there in the market, right? Most of these PaaS, pure PaaS providers in the market, they existed or started building their tech stack pre-Kubernetes states, right? Even including the Azure app service, which is something that I was part of it. Again, it's an orchestrator that is custom. Even Google app engine is a custom orchestrator what not, right? And we saw a unique opportunity for a customer segments in the devs to essentially come and build the application on a PaaS platform that is built using true open source and a cloud native technologies that they are familiar with. And we wanted to make a bet on Kubernetes as a whole ecosystem because we had released the managed Kubernetes offering about a year, a year and a half ago and we wanted to build this new path platform using some of the technologies that our customers are aware of, are understanding in detail and are able to follow their growth journey and innovation as we try and wrap those new innovations, whether they're happening in Kubernetes or other tech that we're using for our platform, and Cody will talk about that, in a cohesive way where they have the flexibility of running the application in a PaaS environment. And if they choose to grow and the application has grown beyond just a simple click and deploy where they want to roll the leaves up, it's the transition back to Kubernetes and the other ecosystem that they are aware of becomes really, really simple. So that's the biggest value prop that we saw besides just serving the customer needs that we've been doing great with our segment that we go after, which is individual devs and SMEs.

[00:05:34] JM: So what is required to build a layer 2 application system? Like what were the building blocks you used to build this system?

[00:05:45] AJ: I'll give it to Cody because he's the one behind the architecting the platform. Cody, why don't you go for that?

[00:05:52] CB: Yeah, absolutely. So first of all it is a big platform. We're serving static sites for customers, we're serving dynamic applications, we're integrating with customer databases. So there's a lot of pieces involved. But as AJ was kind of alluding to, a lot of those pieces are the same pieces that our customers are already working with. We're already building on top of DigitalOcean's Kubernetes offering, our managed database offering our spaces, object storage and DNS platforms there. But in addition to that we also are heavily leveraging the cloud native frameworks. So Kubernetes and then the ecosystem of tools that sit around that. In our case, that comes down to Istio, Prometheus, Grafana, gVisor, Containerd, et cetera.

[00:06:43] JM: Okay. Well, that's a really robust Kubernetes-based stack. Let's go through those things, those technologies one by one and describe how they fit into the overall architecture. So obviously Kubernetes is a container runtime that we've done a million shows on. Maybe you could talk about a deployment to Kubernetes and the – Or we could just talk about a top-down deployment of an application and what the user sees. What's going on underneath the hood? Then we can dive into some of those individual elements, like I'm particularly interested in gVisor and Istio.

[00:07:22] CB: Yeah, absolutely. So one of the things that was important for our system is that it is resilient and robust. And in terms of Kubernetes, it fits in at a couple different players in our system. We have built a control plane on top of most of customers will interact with, and that's where customers actually upload or create their applications, manage their applications, but then we translate that into Kubernetes deployments as you kind of alluded to there. And we run Kubernetes clusters in each of our regions for this app platform and it serves kind of two layers there, where obviously depending on the Kubelet at the base layer to manage the moment-by-moment interaction with the application. Making sure that it's running. Making sure that everybody is getting the experience at their desire.

Above that we're also depending on the Kubernetes API layer to monitor at a higher level to make sure that this machine is up and doing exactly what we expect it to do. And if it's not, it's finding a place to replace that application for the customers so that they stay online and with the availability that they're expecting. We also use Kubernetes as kind of the connecting glue to connect these disparate services like Istio on the ingress side and manage databases as we're controlling the network layer there essentially to facilitate that for customers.

[00:08:49] JM: So can we go deeper into some of the operations that you've had to build around the deployment of an application? Like just when a user deploys, let's say a Ruby on Rails application, what is happening under the hood?

[00:09:06] CB: Yeah. So it depends first on how that user is coming into our system. So we support a couple of different ways for users to provide their application to us. First of all, we will connect with Git or GitLab to download source code directly and we will pull that down, analyze that source code using our build pack system and also some custom code that we've written there. And once we've made a determination about the best way to build that system, we will deploy what works out to be a Kubernetes job to build that software. It's a Docker file. We may use the Kaniko product to build that Docker image. And as I mentioned earlier we also leverage cloud native build packs if we don't see explicit Docker file there to build that customer code. And that lets us dive in and really understand that application at its native level.

So once we've built that source code we will upload it to DigitalOcean's container registry product, and the container registry is a Docker format image or a Docker format registry for hosting those images. At that point, and I should say also here we will also allow customers to upload images directly or reference images in the public cloud via Docker hub. So once we have that image built, that is the branching off point for deploying that image. And here kind of comes the basic parts of Kubernetes. We create a Kubernetes deployment for the customer, find the right cluster to where we can best host that application. And what Kubernetes do, it's doing it's great at in terms of rolling out that deployment in a safe and responsible way for the customer. And then once that deployment is rolled out we begin all the processes of monitoring that application, deploying the ephemeral services around it in terms of ingress and all of the parts of infrastructure that customers don't enjoy working with, DNS, et cetera.

[00:11:07] JM: So once that application is running, what kind of instrumentation do you have around it to ensure uptime and monitor it? And you mentioned Istio. You mentioned gVisor. Let's go a little bit deeper into those technologies.

[00:11:22] CB: Yeah, absolutely. So we are monitoring applications via health checks throughout their life cycle of that application. And so that basically comes down to pinging the application and making sure that it's up in the way we expect. And if it's not, triggering automated actions to get that customer's application instance up. I think another thing to point out there is that we do provide for customers to have high availability in terms of this. And that in that case we make sure that their application is scheduled on multiple instances and make guarantees in terms of any types of rollouts or deployments to make sure that those applications aren't impacted by the replacement of a single instance.

In terms of the ingress layer, you did mention Istio there. We're largely using Istio in a kind of vanilla sense for the ingress gateway portion of our software. We're less using some of the cluster mesh architecture there. But that ingress piece, it provides a good amount of information for us in terms of, again, making sure that those requests are being serviced. And then from there, reaching out to our cloud or to our CDN and ingress support system around that.

[00:12:39] AJ: Yeah. Cody, don't we use that for the monitoring and alerting things like the Fluent Bit Prometheus to provide the data story or at least internal –

[00:12:48] CB: Yeah, absolutely. So we're monitoring applications, application metrics by means of Prometheus and exporting those into our centralized system and exposing them up to customers. At present those metrics are limited kind of to the infrastructure level metrics of the application. So that would be things like CPU. We can provide insights to customers in terms of how much CPU they're using, memory utilization. The type of request that they're getting in and the response codes that we're delivering back for those. So customers can get a sense of what the latency of those requests look like as well as the character of their response. Was it successful or was there an issue encountered and serving that response?

[00:13:34] AJ: Yeah. And again, just to brief up on the gVisor point that Jeff has been asking about. Like that's super critical to our strategy, right? Because when you build this multi-tenant, in a way it's a hostile multi-tenant service where Kubernetes is acting as your orchestrator. You essentially are at the defense of default security isolation that Docker provides, right? In the end of the day if you run those containers on the same VM, the kernels will share. So you always have to have some sort of a security isolation to protect not your customers, but as well as your infrastructure overall, right? And I think the technologies like gVisor made that super easy and possible to build these multi-tenant Kubernetes clusters across the world that are deployed across the world and for us to build the service on top of Kubernetes open source system, right?

And going back to building this custom orchestrator, which are not Kubernetes-based, which is where the most of these existing PaaS offerings are running, right? They all either have no security isolation. They rely on the Docker one or have built a custom one, right? So I think bets on a gVisor, Celium were really, really critical for us to make a decision, to make the bet on Kubernetes to build this platform. And Cody can provide more details on how to use that.

[00:14:54] CB: Yeah, that's absolutely right. GVisor in this case provides an isolation that sits somewhere kind of between the operating system in forest isolation that has been in use since mainframes back in the day and virtual machine isolation, which is the foundation of cloud technology. So essentially what gVisor is doing there is it is running as a micro kernel inside of the operating system that we're running on our Kubernetes nodes. And the advantage of that is that virtual machine isolation is really rigid. The allocations to those guest applications are very firm and there's not a lot of insight from the kernel that's running those applications into what they're doing and how best to serve them.

So gVisor runs in the context of that host kernel and provides nuanced information to the host operating system about what resources the underlying application is using, but still provides – It is itself a micro kernel and provides a lot of those guarantees that – Security guarantees and isolation that we're familiar with from virtualization.

[00:16:14] JM: So we just did a show about Cilium, which I believe is lower level technology for monitoring your network and network programmability. Can you give a little bit more color on what Cilium, and I guess gVisor, if gVisor interacts with it. What functionality do these provide to you yeah?

[00:16:37] CB: Yeah. So gVisor, which we just talked about, primarily provides security on the runtime side. Cilium is a CNI plug-in, so a network interface plug-in for Kubernetes, and we use it to form virtual networks essentially for customer applications and also to mediate the interaction with that ingress. So mediate the software-defined networking Istio and other things that the application needs to reach out to. So for example, manage database services.

And so what this allows us to do is create software-defined networking directly on those clusters that define patterns for securing customer applications from each other and then also help mediate those connections to other services and those inbound connections from our customers' customers. And I guess to speak to a bit of the technology that it's using under the hood, it sounds like you've already spoken with the Isovalent team that makes Cilium. Under the hood they're using BPF. So byte code that runs essentially in the kernel and provides a more nuanced way of executing those firewall rules is essentially what they come down to be than the kind of traditional IP tables way and also more performant because of that.

So when you think about building the app platform that you've built for DigitalOcean versus what you would have had to do, let's say, five years ago, pre this whole Kubernetes ecosystem, how does it compare? What would you have had to do back then that you don't have to do today?

[00:18:24] CB: Yeah, that's a good question, Jeff. So I can speak to – I've had had past experiences working in PaaS type applications as well. And you're right, that there is a lot of benefit that we're seeing from this Kubernetes and cloud native ecosystem. For example, the two things that we just discussed, the runtime interface. Traditionally, something like that would have required working with very low level Linux primitives. And in doing that, also having a great deal of confidence in the – I think the advantage to having those in the open ecosystem there is that they are well-tested and also the scope of those products is well-understood. And for a product like the app platform where we're interacting with customer applications from every flavor I think is an important thing to say there.

Having the scope of these well-tested tooling really enables us to provide services that otherwise might not be able to there. In terms of Cilium and some of those networking primitives there, again, providing those in the past might have been the sole business of an entire company. So there's a norm. You can think of Open vSwitch and some of those technologies, and those are still in the open source realm. Building technologies like that I would have been at the scale of a Cisco or something along that line, and that's their sole business. So providing or these in the open ecosystem enables us to focus on customer experience and then also to really – And we can dive into these and understand them and change them as we need to, but they come to the table with a lot on their own.

[00:20:10] AJ: Right. Right. To Cody's point, the time to market has change dramatically with these open source technologies allowing you to build this orchestrator, right? I mean, just look at Kubernetes, right? I mean, this sort of a platform probably needs a single cluster. Probably back in the days where you need about 1500, 2000 VMs. Orchestrating the health of those VMs, if some customers application goes down on that VM, moving to another one, storing the state. It's in cluster one or cluster two, right? I mean, all these things that we had to think about five years ago, you get them for free with Kubernetes, right? And talk about ingress, right? Having those load balancers in front of these gigantic clusters. And as they grow, you keep on adding them and whatnot, right? And building this logic and which stores what and what not, right? Again, Istio and things are around the different ingress that we have in open source makes these lives really, really simple.

And not to forget the CDN that we use. We partner with Cloudfare, right? The largest CDN provider in the world with awesome benefits that we get around the SSL creation, the easy integration with things like less encrypt, right? Managing your DDoS attack, right? I mean, just imagine trying to incorporate DDoS attack. Algorithms within your loan balancers back in five years or whatnot. I mean, these are the things that were super complex and difficult to build a system like this without the innovation that we've seen in the open source ecosystem around the world. So we're super excited and thankful to that.

[00:21:40] CB: I think one thing is this helps us build from the experience that every – From the world's worth of experience. So as anyone who's run infrastructure can at scale can tell you things go wrong that you haven't predicted and haven't had experience with at size, and that will keep happening. So leveraging these technologies allows us to build on the experience of those, of everyone in the world who's had those experiences and benefit from their pain in some ways. That way our system can handle these things in an automated way as opposed to being a surprise.

[00:22:24] JM: In what ways has the building out of the app platform been more technically difficult than you anticipated? Or what were some of the most technically challenging parts of implementing it?

[00:22:42] CB: I think one thing to speak to on that question, a good one, Jeff, is that we are deploying varied customer applications with this. The variety of customers and the things that they want to run in our app platform and in our cloud is enormous. We have people using every different framework that you can think of in every different type of custom application. So anticipating all of the ways that customers want to run their application makes for an interesting challenge. And I think we've done a good job at finding the balance between having opinions about the right way to run your application and also trying to meet customers where their application is already. But that does make for a lot of unique challenges.

[00:23:30] JM: Anything to add there, AJ?

[00:23:33] AJ: I think Cody added to the right point, but when you look at the technical challenges, obviously there are tons of them, right? But then tons of things that we would have faced otherwise we're available in open source. One of the existing challenge and the one that I foresee for a lot of these PaaS providers in a – Especially people building on top of Kubernetes ecosystem is a shared storage, right? At the end of the day, not all the applications are just microservices, right? Or there are some still require statefulness. And if you were to take an example of the largest web application that everybody uses is WordPress, right? I mean, the persistent disk becomes super super critical, right?

And understanding how do you solve for those scenarios in a multi-tenant large clusters where customer containers can be running on a scale from 1 to 100 trying to access the same storage, or might be going down at any time trying to do a write versus read. And how do you go about solving and enabling those scenarios, right? And build that shared disc or persistent storage that works across this multi-tenant systems is a challenge and it's a really big one, right? Yeah, you could go out and easily plug in to some of the central object storage and try and treat them as a file servers or have a big NFS servers, but trying to maintain those states while the translations and making sure the data corruption doesn't happen. Making sure the performance is amazing, right? Because most of these technologies do live in cloud and might not be sitting next to where your containers or compute are, and how do you handle the caching of reads and writes? I mean, these are some of the really, really complicated challenges, right?

And I’m sure even in the Kubernetes ecosystem people are trying to solve these in a different ways. I do see tons of startups trying to solve this one way or another, but we haven't found the right answer to start supporting those sort of scenarios where you really need a persistent disk. And that is accessed across multiple compute instances on demand and with a variety of workloads. I mean, that's always a huge challenge and it's something that we kept on exploring. And we did discuss quite a bit back in the days when we were building this and it's also an opportunity for us to do some innovation moving forward.

[00:26:01] JM: So you're saying shared persistent state is something that is particularly difficult?

[00:26:07] AJ: Not the state, but the disk, right? The data where you – Actually, the content. Like the WordPress is a great example, right? A typical WordPress site will have thousands and dozens of directories that are nested, and the content every time you're uploading image or writing blog or whatnot needs to be in a central place even though it's the largest hitting site or whatever. Your application might be running on a 10 compute instances, right? All 10 of them trying to access the data that is persistent and latest update, right? All 10 of them trying to write to the central place can be a challenge.

[00:26:46] JM: Got it. So you're saying if you had 10 WordPress instances that we're trying to write to the same –

[00:26:54] AJ: Remote disk, yep.

[00:26:56] JM: Yeah, the same disk-backed file system.

[00:27:01] AJ: Yeah. Yeah. Yeah. I mean, your website, if it's large, even though it's WordPress, it can be hitting a lot of traffic. You need 10 instance, 9 instance, or whatever. It depends on your traffic. But data needs to be consistent, right? And now we're building these cloud native applications. There are different technologies. There are microservices at the end of the day. And having to store that state in a meaningful, in a cheap and less expensive way to the end user is a challenge. I haven't still seen anybody really solving that in the phase of a Kubernetes market or ecosystem all up yet.

[00:27:37] JM: Have there been any other ways that the user base has pushed the functionality of the app platform that you've built? Like ways that users have unexpectedly broken your past platform have had to change it?

[00:27:57] CB: I think one of the places that we see a lot of behavior that we weren't necessarily expecting, I guess would be the right way to say it, is in terms of customers who actually have some experience managing infrastructure of their own and have to a certain extent build patterns around that that they're hoping working. And sometimes those patterns are great, but sometimes those patterns actually kind of butt up against the managing of infrastructure that we're trying to do for the customer. And so an example of that might be customers who are trying to install monitoring at low levels of their application. So they're trying to themselves monitor the amount of data that's flowing in and out of their network interfaces and monitoring the CPU utilization and memory utilization at layers that we're trying to protect the customer from having to manage the infrastructure at that level. So that's been a challenge in some ways that we weren't expecting because our mind was that we were coming into this to help customers avoid those challenges in the first place.

[00:29:06] JM: Interesting. So the customers wanted to go lower level than you were trying to implement the platform for.

[00:29:16] CB: Yeah, absolutely. I mean, we're trying to design – We have a platform that is designed to run container instances, but a lot of our customers, a lot of our existing customers are coming from a virtual machine kind of mindset. So helping customers get accustomed to that has been a surprise I think that we weren't necessarily expecting. But I think once customers kind of have found that those concerns are outside of their – They no longer need to have those concerns. It's been a good experience for them.

And then on the other side of that we have a good number of custom – I mean, the app platform was built for customers to bring in their source code directly and not have to worry about infrastructure at all. And so for those customers, a lot of them are leaning pretty heavily on the build pack support that we have there. And to a certain extent, I think our expectation was also that customers might leverage Docker files when they hit a problem. But being able to support those customers with build packs has been a really good experience when things just work for them.

[00:30:25] JM: This might be a question more for AJ. There is such a proliferation of PaaS platforms out there today. What's the competitive differentiation or the competitive positioning? How do you try to make a PaaS platform that's different than the other ones out there?

[00:30:43] AJ: Yeah, that's a great question, right? Again, the number one thing that comes to my mind is essentially what we do here at DiO the best, right? Our company ,the product, the positioning is all evolving around the simplicity, the community and the pricing, right? So I think the experience that we provide with the PaaS platform that we have stays true to our DNA with the rest of the product experiences that we provide. It's just outright simple and it's priced to performance, right? So when you start looking at some of the competitive pricing across the larger PaaS platform, this just makes sense. The startups, the individual desks, people trying to get their hands dirty with learning new technology, right? We have a great ecosystem of community. 35,000+ tutorials, right? It's a great way for them to come in and learn and get started with something really, really easy. I mean, that's the number one differentiator that we see.

The number two that I can talk about is, again, it goes back to some of the technology bets that we made, right? Even though it's an opinionated platform by definition, any PaaS is an opinionated platform, right? The underlying technologies that we use is they're all open source. So all the innovations that are happening within those ecosystem, we try and capture those within our PaaS offering. And you as end customer feel confident, comfortable about all the innovation that you see in there, out there and open, how they are moving and whatnot. So this whole notion of lock-in is relevant to you, right? This thing was an app platform and it's running on top of Kubernetes, right? At some point if you grow large enough for you to abstract out of it, you can literally use the same stack that we're using and your application will run, right? You really don't have to rewire and rewrite your application because it was written to work on a PaaS platform, A versus B versus C in that perspective, right? So I think that's a huge advantage.

And number three, again, we're a pure play cloud provider, right? So now going back to running all these uh software on top of our own infrastructure, we have better margins. We control the cost. So we're always able to pass on the better pricing back to our customers, which some of these pure play provider might not be because they are leveraging AWS cloud or GCP cloud or something else to power that experience, right? So for them, the price to performance, the price of value will always be a difficult battle to win per se. I think these are three strengths that we see within the ecosystem that we built. And plus, again, you're just not a pure PaaS, but you're not just coming in here for PaaS. As your application grows sometimes you need a broad VM to do certain other processing that a PaaS provider doesn't provide. And you need managed databases, right? You need firewalls or maybe different kind of droplets, right? You get a whole set of core products here within the single cloud. Where with the pure PaaS play, you might have to go and do different things in a way with – One part of the application might be residing in a different cloud while you're doing this application or web hosting with a pure PaaS place. I mean, these are key differences and unique opportunities for us.

[00:34:05] JM: So you've mentioned you have a monitoring stack that's based around Prometheus. I'm wondering how you monitor all these apps at scale and how you surface anomalies or problems that are occurring across these apps when you have, I assume, thousands of applications running. What's your strategy for error discovery and resolution?

[00:34:32] CB: Yeah. I think the most important thing there is that this is obviously a tiered approach. We have at various layers of our system both that alert monitoring in terms of Prometheus, but also monitoring from the Kubernetes stack itself and resolution from the Kubernetes stack itself in terms of some of its self-healing properties of rescheduling away from damaged instances and restarting applications when they encounter problems. And then we, as you mentioned, leverage Prometheus in those clusters. And then we leverage some of Kubernetes federation properties to filter out the metrics that we think are important up to a Prometheus instance or to many Prometheus instances that we are running behind the scenes within our data centers that are then able to bubble some of those metrics back to the customer. And Thanos is another bit of technology that we're using there, which just essentially allows clusterization of Prometheus.

[00:35:40] JM: How do you structure teams around a platform like this? Like do you have DevOps teams? Do you have do you have like different product teams? I'm just wondering about the management structure and how you manage it? How you grow it? Just the strategies around the management.

[00:36:03] CB: We have a core app platform team which is responsible for the operations side as well as the development side of this. So we are a true kind of DevOps style organization there. But DigitalOcean also has a kind of top level operations organization that is that kind of first line in terms of operational defense in terms of managing and keeping a reliable system for customers. I'll let AJ speak to the product side of this. But, again, on the engineering side, we're also integrating with an enormous number of different teams within DiO. So we had a team that was specifically focused on the static site side of the app platform, and that's something we haven't touched on a lot in this call yet. But it is an important part of the app platform. Somewhere we worked with our managed database team to get the integration right with that. And then, again, container registry. So that's another part of this organization.

So I think a thing to say there is that we are building this out of a lot of the products in DiO and we're able to leverage those products, those teams. So it was all DiO and all DigitalOcean effort to make this work correctly.

[00:37:23] AJ: As Cody said, I mean, we're very methodical about this, right? This question really brought me back a year and a half or a year ago when I had a chance to essentially pick the team and build it from ground up in a way, right? So we do a lot of things here at DiO besides the app platform, right? And one of my biggest criteria was I wanted to pick the right and the best to build this technology based on the architecture that I had online, right? So we essentially made the bets on people who were part of a team that built the Kubernetes, the managed Kubernetes offering, right? I think Cody was one of the members and Stephen as well, right? And the idea behind building a team that have built the core orchestrator on which we're going to build this layer of abstraction was super critical, right? So that's number one philosophy about going and building that new product was to have somebody who've done the core experience.

And then less cook in the kitchen. The better it is, right? So we try and keep the team lean and mean. So we're able to execute faster and invest in things that are meaningful to our end customers. And the DevOps, again, we’re different – We organize in a different product line, right? If you just look at droplets, which is the core VM business, we do have a dedicated DevOps team or SRE team because it's heavily involved in infrastructure, whatnot. But in the software, the philosophy that I like is that DevOps usually works, right? I do want people who wrote the code to get woken up in the middle of the night, because I know if that's what happens, next day morning, what are they going to do the first thing is to go and fix that, right? Versus waking up somebody else who probably don't have an idea or haven't introduced the bug that they wrote. So I think that's super critical for anybody building a software or this sort of a large scale service is to not to overly optimize for protecting the dev, right? As the platform becomes bigger, as the team grows bigger, I mean, you could take – And as your knowledge base gets bigger, you can take this approach. But having DevOps within the devs is really critical, right?

So yeah, we assembled a team of a core people who had worked on these core technologies, the Kubernetes team, the people who built the registry, the people who were working on a static site with us, object storage and CDN. We brought them in and built the core team, and that's what the engineering team looks like. They do the DevOps. We do have a support team, but we have a strong culture within our engineering product to essentially go out in community and answer people's question. And that's really, really critical and vital, right? Because if you rely on support, those question and answer gets stored into your closed system, whichever system that you use for ticketing. And that data is not out there.

And when you're building a product that is developer friendly and focus on the devs and essentially the issues that you run into are more application-specific because your system will be here when the way your application is written, all these questions and answers, the more they are out in community, the better it is. So another big part of the dev team and product team job is to go out in community and try and answer every single question that is out there. And we have our devs answering those questions, right? We rely a little bit on support where people go through and like to get a support that way. But mainly most of our troubleshooting with our customers happen in community.

And from a product perspective, again, this organization, I run the entire product team here at DiO. So there are different product portfolios. Our platform sits into what I call is a PaaS product portfolio where we have app platform, managed Kubernetes, managed databases, the marketplace, and it's essentially led by one leader who has a couple of product managers who work with Cody and his team. And out of those two product managers I have one focusing working very close with Cody and his team who's super technical as well as a product person with a vision on the direction as well. While one other product focuses on a typical product management around the business side, the customer acquisition, how these technical investments are turning into business insights and improvements and how do we build bridges with different technologies to be used within the company.

[00:41:41] JM: What's your vision for the future of the app platform? How do you improve it? If you already have the ability to deploy a wide range of application types, you have scalability built in. What are the places where there's room for improvement?

[00:42:00] AJ: There are tons of places, right? And by no means we're feature complete, and you never will be, right? I mean, as your customers come in – To Cody's earlier point, we're getting customers who are making transition to PaaS and they come from the IaaS world and they use product in a different way, which is absolutely fine because that's the opportunity for us to go out and to build a feature and to make that experience seamless, right? But besides that, there're tons of ways to optimize, right? I mean, each and every pieces that we talked about starting from the ingress, from the actual container hosting, the file server that we briefly talked about, the load balancer, how we build the customer application I mean, that in itself can be entire CICD pipeline and providing insights into those builds, right? Providing additional insight on how your containers are working. Maybe going after a non-web workload as well at some point and a data workloads, right? Row right now they run on top of Kubernetes, which are using our droplets once we support GPU in future, having the PaaS for AIML.

Obviously making bets on this open source technology allows us to innovate much, much faster, like serverless, right? That's the next big thing. And there are tons of opinions around like how the function as a service should look like, right? People are still trying to define what that should look like, whether it's AWS Lambda or Azure function. Again, something that I had a pleasure to build back in Microsoft, or the direction what the Cloudfare is doing, right? Which is the function should be on the edge, right? But we're well positioned to write the wave and help drive the innovation in the direction whether we're going to build the serverless on our platform. We have the technology that is Kubernetes running on top of it, right? We have Istio in place we know all the bells and whistles how it works. And literally turn on event-driven backend base, the functional service within a few quarters. Or we choose to say, "Let's provide the edge functions, like the direction in the Cloudfare or the Netlify's of the world going. We already are in partnership with Cloudfare, and our platform works great with their CDN. And essentially providing functions on the edge. And that also becomes an option. So there're tons of scenarios and tons of value prop that we can add and build this platform on. There's a reason why we called it as a platform versus a single product per se because that's essentially what we're building. And where that innovation is going to go is we're going to listen to our customers, hear from them and see what they want and keep on evolving in that direction. But this is truly the future on how we see people build applications moving forward versus rolling up the sleeves and spinning a bunch of VMs. Don't get me wrong. People still do that especially on startup where they're cost sensitive, trying to save money and are technically skilled. But most of the developers, the next-gen devs, the citizen devs of the future, this is where they're going to build.

[00:44:59] JM: And given that you both work at a infrastructure company. I mean, you literally work at a cloud provider. There's not very many layer one cloud providers. I would love to get your perspectives on the kinds of infrastructure changes you think we'll see in the near future. Like serverless, for example, I’d love to know if you guys have any firsthand insights into do people want more out of functions as a service? Like I’m not sure, DigitalOcean doesn't have a functions as a service platform, I don't believe. But I’d just love to know your beliefs about the future of infrastructure more broadly and what you think we'll see in the coming years.

[00:45:45] AJ: Yeah, I can take that question, right? Again, we don't have the function as a service yet, but we have the platform ready when we are ready to execute on delivering that for our customers. Infrastructure as a service or IaaS in general, right? I mean, it's here to stay. Let's be honest. We've been talking about the death of IaaS for the last five many years or so, right? It's still here. If you look at the balance sheet of any large cloud providers, right? What makes up the money? It’s still VMs. They're all storage and things along those lines.

So in my opinion it's not dying. It's not going to go away. There is enough of cloud workload that needs to be moved or the on-premise workload or the hybrid workload that needs to be moved that does need beefier and a higher VMs and a raw droplets per se. And that's here to stay. It's not going to go away. But most of the greenfield apps, the new web applications, the individual devs, the smaller startups who want to start and prototype something quicker and don't want to overbuild from day one, that's where you are slowly seeing the transition into the PaaS and the serverless of the world, at least on the function side of the things.

So I think the infrastructure business all up will continue to see the innovation. Those innovations are going to be more heavily focused on bringing the optimization. Doing more with the less versus just the outright, the customer-facing feature per se. You're already seeing signs of people trying to do serverless. They're using infrastructure directly with a soft layer of a software versus a full-blown heavy stack of a function as a service, which is event-driven in a way. Like this container as a service is a new trend where you see the Kata Containers, the Firecrackers of the world, right? Where just giving out a container from the raw VM with a single API, that's one recent bigger innovation that you've seen. But the infrastructure business all up, I would say it's here to say. There's enough appetite within the larger or smaller customer segment to roll up the sleeves and build the stack out if they choose to and if it makes sense.

Again, there's a gradual transition around like from CPU to GPUs as people start using more AIML and get that mind share. Like the innovation that we see in the GPU stack and the GPUs cards that you're able to run within the server and how do you try and do a multi-tenancy with that and take that in that journey? It's super critical, because in my opinion I'm convinced at least on the infrastructure side the AIML technology five years from now is going to be what databases are today, an essential part of any application. It will be just taken for granted. It's not one of those special case scenarios. You're going to need it just like how you need databases.

And on the pure software side on PaaS perspective and functions, but not, again, those citizen devs, right? The abstraction of a workflow on top of that. Most of our kids are learning how to code in school, right? Wow when they're back in the workforce, not all of them are going to be devs, but they're going to use those skills to augment their core job. And building those citizen apps and layer of abstraction is going to be critical. But all of those, whether it's a function as a service, PaaS, citizen app, they all require innovation at the bottom layer of infrastructure, and that's what I meant by the optimization, the efficiencies on how that bigger picture gets evolved is here to stay for a foreseeable future.

[00:49:32] JM: Cool. Well, that seems like a good place to wrap up. You guys have anything to add?

[00:49:37] JM: Oh, nothing on my side. I think it was fantastic. Yeah, thank you for asking those questions. It was really, really useful.

[00:49:44] JM: Okay, great. Well, thank you both for coming on the show. It's been a real pleasure.

[00:49:49] AJ: Likewise. Thanks. Take care.

[00:49:50] CB: Thank you, Jeff.

[END]