EPISODE 1265

[INTRODUCTION]

[00:00:00] JM: Running applications in containerized environments involves regularly organizing, adding and replacing containers. This complex job may involve managing clusters of containers in different geographic locations with different configuration requirements. Platforms like Kubernetes are great for managing this complexity, but includes steep learning curves to efficiently get anything off the ground. The company, Portainer, provides a universal container management tool that works with Kubernetes, Docker, Docker Swarm and Azure ACI. It enables managing containers without knowing platform-specific code and best practices. Instead, deploying containerized applications is done through a simple graphical user interface. Once deployed, you can observe and monitor the apps and governance security settings all through Portainer.

In this episode, we talked to Neil Cresswell, a co-founder at Portainer. I hope you enjoy the episode.

A few announcements before we get started. One, if you like Clubhouse, subscribe to the Club for Software Daily on Clubhouse. It's just Software Daily. And we'll be doing some interesting Clubhouse sessions within the next few weeks. And two, if you're looking for a job, we are hiring a variety of roles. We're looking for a social media manager. We're looking for a graphic designer. And we're looking for writers. If you are interested in contributing content to Software Engineering Daily, or even if you're a podcaster, and you're curious about how to get involved, we are looking for people with interesting backgrounds who can contribute to Software Engineering Daily. Again, mostly we're looking for social media help and design help. But if you're a writer or a podcaster, we'd also love to hear from you. You can send me an email with your resume, jeff@softwareengineeringdaily.com. That's jeff@softwareengineeringdaily.com.

[INTERVIEW]

[00:01:51] JM: Neil, welcome to the show.

[00:01:53] NC: Great to be here, Jeffrey.

[00:01:55] GP: We've done lots of shows about container management. And in particular, we covered the container orchestration wars back in 2016, or 2017, whenever that was. And I thought that basically, the wars were over Kubernetes was the container management platform to use. And then you had some options on top of that, like you could use standalone containers from Amazon, like Fargate. Or you could use the ones from Azure, like the ACI instances. And then there were a few kind of container management companies like Rancher that seemed to be doing okay. But I thought that, generally speaking, the game of container management was over and the opportunities for platform companies built around container management we're gone. You have a company built around container management. So I'd love to get your perspective on the market and why there is an opportunity for a container management company.

[00:02:56] NC: Sure. Well, the war, so to speak, is definitely not over. There is also still a stronghold of Docker Swarm lovers out there. We still see new users of Docker Swarm hitting Portainer every single day, which is surprising. So there are still people choosing Docker Swarm versus Kubernetes. So that that war is still ongoing, albeit Kubernetes is definitely out leading the charge there by a country mile.

In regards to managing the container platforms, to be completely frank, that war is only just begun. And again, that is not necessarily even a war. That's more, how do you actually corral these relatively unwieldy things called containers, because they are so hyper dynamic? If you think back just a few years, people were really worried about VM sprawl and having virtual machines basically sprouting up like mushrooms, because it was just so radically simple to spin up a virtual machine. Well, this, with containers, is actually multiplied, I don't know, 100 fold, if not more. It's so monumentally quick and easy to spin up a container that organizations without knowing may go from having a couple of containers as a test case or project to having many hundreds or thousands of containers without any real effort, because it's just so simple and easy to spin them up. That makes it really difficult to start to scale and protect and control your environment when you've got these applications that can be spun up in seconds, connected to your corporate network and have access to everything in there. How do you know who's spinning up applications? How do you know that what they're doing is not privileged or

protected or in any way hostile? Every organization potentially has one hostile actor within their company. How do you ensure people are adequately controlled and managed and governed? So there're still a lot of elements of container management that are yet unaddressed by any of the other platform providers out there.

[00:04:56] JM: So given your knowledge of the wars, and these different companies that you've seen built around container management or the different products? Like another one I forgot to mention, OpenShift. There's a bunch of these. How do you differentiate? What do you do in terms of container management that's different than these other platforms?

[00:05:19] NC: So we've really focused on the consumption layer. So there are two elements in regards to managing containers. There's the creation of the platform, and there is the consumption of the platform. And the creation of the platform is like building the clusters, adding nodes, installing the container runtime, installing the orchestrator, any storage drivers, network drivers, all that kind of stuff. That's what people like Rancher, and OpenShift and others do very well. If you want to self-host, your Docker or Kubernetes environments, you can use one of those tools that will bootstrap your cluster with all of the recommended configurations and give yourself a cluster. We think that's going to become quite marginalized, because the cloud providers let you do that with a credit card with the managed Kubernetes services especially.

And so we're focused predominantly on the consumption layer. So helping an organization's users consume the capabilities of the cluster. And by that, I mean, make it really easy for them to deploy applications and manage those applications post-deployment. So our sole focus is on providing a natural language interface that users of any stage of understanding in regards to container management can log into Portainer and deploy applications without necessarily needing to know much, if any, of the underlying technology architecture. So in Portainer, as example, user can login, deploy an application with even the most complicated configurations on top of Kubernetes. And they don't necessarily need to know anything about Kubernetes.

[00:06:56] JM: And who's the typical customer? Is it like a legacy organization that's refactoring or are there like startups, or like mature startups? Who's the typical type of customer?

[00:07:11] NC: We actually see A-B customers. So A customer is a startup who is just getting going. They haven't yet had the time to invest any money heavily in automation, or they don't have the knowledge to invest in automation. They just want to build their application. Get it running. And predominately, these are either low-code developers or frontend developers that are using frameworks that don't require them to understand much about the underlying infrastructure. So those people want to just use Portainer to deploy an application. They would use our application template functionality a lot, because they can just spin up a database server and a web server and start throwing their application on it.

The other side, that B customer, is actually the larger enterprise who have a very, very significant investment in people in their developers and/or operations staff who cannot afford the time or money to retrain their entire fleet of users. So if you think of an organization worth 500, 1000, 2000 developers, if those developers are not suitably conversant in microservices, containers, Kubernetes, how do they transition their application development from a legacy, more monolithic type architecture to micro services without significant retraining, which comes at quite a cost? So Portainer, in that case, helps them lower the barrier to entry so that their existing staff can get going and using this technology without necessarily having to give them prior training. And we've seen that quite regularly across our very large users.

[00:08:55] JM: What what's your perspective on the current Kubernetes ecosystem, like in terms of market penetration? How is the technology evolving? Is it becoming easier to use? Give me your perspective on just the overall ecosystem.

[00:09:14] NC: So the ecosystem is simply massive. I don't know if you've seen the CNCF landscape, but you can probably wallpaper a wall of your house with it. It’s just simply staggering how many organizations are part of that landscape, which is great if you're really into choice. But if you're into just getting things up and running, it can be quite overwhelming. How do you know which is the best network driver to use? How do you know which is the best storage driver to use? How do you know which is the best tools for monitoring your containers? How do you know the tools, the best tools for building multi-cluster applications? There's just such a vast array of options available here. And options are great, but also options can introduce significant confusion. And that confusion can simply be a barrier to entry for so many

people. They just don't know where to get started, because there's just so many things to choose from to get started.

So at this stage, I would say it's still a very, very complicated environment to live within. If you just want to get started, get working with Kubernetes, it can be quite daunting. And again, there are there are a number of tools like Rancher and others that let you spin up an opinionated version of what they deem is the best practice componentry. So you don't need to go and hunt around everything in the landscape. You can just go and use the best of breed components to build your cluster. But still, if you don't necessarily know what you're doing, it's quite difficult to basically log in and start deploying applications. You really do need to know the architecture of Kubernetes, the architecture of how load balancers work, or Ingress controllers work, or persistent storage. If you don't know that, it's very hard to get started. So it's still a very, very confusing time.

And I think that's why adoption of containers and enterprises is still quite low. I saw a survey recently, which sampled quite a large number of users and said write the number of the applications that are containerized in your environment, and it was about 5% of corporate applications were running in containers. So if you sample the echo chamber and say, “Everyone running containers, how many of you are running Docker or Kubernetes?” You're going to get a very large number. Or what percentage of your container applications are in production? 80%. And everything's, “Oh, cool. 80% of our applications are in production.” But this was one of the first surveys I saw which said of all of the applications running in your organization, how many of them are actually running in containers? And it was as low as 5%, which shows that we've still got a very long way to go before we get mainstream adoption across all applications that power enterprises today.

[00:11:52] JM: Has the typical onboarding process for an enterprise migrating to Kubernetes, has that changed over time? Or I guess I should say migrating to containerization? Has it changed over time? Or is it the typical strategy, still, you kind of get some minor part of your application on to containerization and then you gradually eat up the rest of the stack over time?”

[00:12:21] NC: I don't think it's changed a lot to be honest. Stateless containers, containers, they don't actually store any part of the configuration locally. Those are still the simplest form of

container to get up and running. It's also the simplest form of application to convert into a container. So I think organizations are still starting with that approach of saying, “Well, what of the frontend elements of my application or even the middleware, web servers and API gateways are almost always the very first things to be converted into containers?” because they can store their state externally in a database or some other mechanism.

When you come into the middle layer, the heart and shell of your database servers or for your applications that actually require state to be held, that's where things start to get really complicated, because all of the dynamics of storage performance haven't gone away. So when you're running things like a stateful database server in a container, you still need to think about, “Well, how am I going to get the database stored somewhere in my cluster that has high-performant? That has got high IOPS and high throughput and is highly available? Can be shared across all nodes of the cluster? That gets very, very complicated. And there're only more mature organizations that would likely embark on that journey.

So I think it's still relatively a staged approach, crawl, walk, run. And you start off with a very, very simple front end application. You might be in transition to your middleware, or things like your message bus, RabbitMQs or something like that, something relatively stateless. Once you've mastered that, you then move on to your stateful apps. That's how I see things still progressing anyway.

[00:14:03] JM: Tell me a little bit more about the architecture of Portainer.

[00:14:07] NC: Okay, so it has a back end component, which is you can call the Portainer server. That is a Golang application that runs itself as a container. And it can run either in one of the clusters that it's managing. Not recommended, but you can. Or you can run it on a dedicated virtual machine or a dedicated container platform standalone. Call it your management cluster. And it will then connect to and manage all other clusters that you need. Users don't install anything on their PC. Users access the Portainer instance through a web browser. And we have an Angular frontend, which runs in the browser, connects to the backend and all of the activities are from the browser to the backend. So there's nothing installed on the user's PC. There's no need for conflict files or kubctl or anything. It's all browser-based management.

Portainer is all locally hosted by the user. It's not a SaaS offering. So the user can run it inside their secure network. It doesn't require any Internet access. It's completely isolated environment. Portainer itself manages all of the clusters using our agent. So in the clusters that you want Portainer to manage, you install a little micro piece of software, which is a Portainer agent, and Portainer agent connects securely to that Portainer container for this remote management capability.

We have a version of the agent that is designed for edge compute environments. It basically is powered by a reverse tunnel technology. So you can have the edge agent running on Internet connected Kubernetes or Docker environments out there in the wild. You don't need any port forwarding or net address or anything. As long as the remote devices can connect back to the Portainer instance, the edge agent will establish your reverse tunnel to Portainer to still allow interactive management of those devices over the public Internet.

[00:16:06] JM: So do you expect the average user to have a Kubernetes cluster or to be using standalone containers? Or do you require them to be using – Like what's the spec for what you require them to be using?

[00:16:22] NC: So we are actually completely agnostic. We're all about helping users deploy container native applications. We don't actually stipulate any orchestrator under the covers. So if you want to use Docker standalone using Docker desktop on your PC, go ahead. If you want to use a small Docker Swarm cluster because that's what works best for you, go ahead. If you want to use a really lightweight Kubernetes distro like MicroK8s. Go ahead. If you want to use a full featured heavyweight Kubernetes environment self-hosted, go ahead. If you want to use a managed Kubernetes service from a cloud provider, go ahead. We're not actually in any way, mandating how your environment is deployed or configured. As long as it supports Docker, Docker Swarm, Kubernetes as your ACI, and in the future, a range of other serverless container providers. Now then we will let you manage it. We're not a Kubernetes dashboard. We're not a Docker dashboard. We're a container native application tool. So yeah, any online platform is fine.

[00:17:27] JM: Give me a little bit of description for what a container management solution or Portainer specifically. What problems does it alleviate or make easier to manage?

[00:17:39] NC: So spinning up a container can be relatively simple. You can spin up a container just with a simple Docker run command. And it's up and running. However, how do you know that that container is running in a way that is safe and secure? How do you know that it's compliant with best practices? How do you know that it's not running privileged? How do you know that a user has not elevated the permissions of their container to have access to things like physical hard drives in the Docker host? There are a whole bunch of capabilities that users have access through the container orchestrator that they may or may not need in the day-to-day operations of the business.

Why would a regular user be able to start a container that can bind mount HD zero, which is the actual block storage device on the Docker host? What is the reason to do so? What Why would a user need to be able to bind mount the root filesystem of the Docker host? Why would they need to have access to physical devices? Why would they need to run as root in a host? There's a whole bunch of considerations when you're deploying applications that things like Docker and Kubernetes don't get in the way of. They simply say, “Here is an open API. We trust you to trust your users. We won't provide any kind of constraints. Your users can do anything that they asked for.”

It's beholden on the user to ensure that the system is configured in such a way that they are protected from malicious acts or even just mistakes. And Portainer’s goal is to enforce these best practices in the frontend of the application. So in Portainer, an administrator can actually disable the use of sensitive configurations. So we won't let users elevated the permissions of the container. We won't let them get access to sys control. We won't let them bind mount to hard drives or bind mount the root filesystem if the admin decides that is a bad idea. So we have a whole bunch of best practice policies that we have a bit into Portainer that controls what the end user has access to from within that organization.

And in Kubernetes, it's things like the default namespace. Yeah, the default namespace is available on an on every single Kubernetes cluster. All users have access to the default namespace. It is in no way protected or secured. And so it can be used as a dumping ground. In Portainer, we say, “Well, actually, that could be quite a dangerous thing.” Every single application really should be name-spaced and should be constrained by some kind of quota or

some other permissions. So we provide an option for an admin to say no one can use the default namespace. Everyone must use actual namespaces. And then with actual namespaces, we enforce things like broidering. So we'll go and turn on a quota and say, “For this namespace, you have this much CPU and RAM and disk and load balancers and ingresses that you can use. No more.” And we basically go and lock and constrain that.

We also do things like protect the organization from out of memory crashes. One of the worst things that can happen in your environment is that too many containers are deployed. They use too many resources. The cluster runs out of resources. The Linux kernel pulls out this 44 Magnum and starts shooting process as a random called out of memory error, which can take down your entire cluster. With Portainer, we give the administrator to set a threshold to say, “Do not allow users to assign more resources to containers than is available in the cluster.” And we let them say – Even reserve 5%, 10%, 15% of the resources for system services so users can only consume the balance. So we have a whole bunch of protections in place to ensure that the system stays up and stable.

[00:21:42] JM: So can you tell me a little bit more about how your control plane works or how you interface with Kubernetes? And like, I guess, how data is fed back between Kubernetes or the containers that you're managing. And I guess the interface between Portainer and the containers themselves?

[00:22:03] NC: Okay, so using the Portainer agent. This runs in the clusters that we're managing. Those talk directly to either the Kubernetes API or the Docker API directly either using sockets or just using the API port from a Kubernetes perspective. So Portainer talks to the agent. The agent acts as a proxy through to the API, and proxies are requests through to the API natively. So inside Portainer, when you're saying, “Give me a list of all of the applications or pods running on this cluster.” Portainer makes the call to the Poretainer agent. The Portainer agent the in proxies the call to the Kubernetes API, retrieves the information, filters out anything that needs to be filtered out from a security perspective and provides that response back to the user.

One of the important things to note is that inside Portainer, we support users and user profiles. So you have a whole bunch of users and teams defined in Portainer. When you create a user

and assign them permissions, we actually then automatically go and create a user on your behalf in the backend Kubernetes cluster. So we'll go and create a user. We'll assign all of the required RBAC rules. And we'll basically configure the back end for you. So that any connection from the user through the backend environment is safe and secure.

[00:23:27] JM: What kinds of instrumentation do you see the average Kubernetes user deploying alongside their – Or I guess container user deploying alongside their infrastructure? Like are you seeing a lot of service mesh usage or usage of – We did a show on Calico recently. Tell me about some of the infrastructure that you're seeing paired with container infrastructure most frequently.

[00:23:56] NC: We might be in our own little micro-echo chamber, because we currently don't support service mesh configurations. So we don't see a lot of service mesh deployments. Now that may be just the fact of the matter, or maybe that our users don't use it. What we see definitely is a high-degree of Ingress use. Traffic is very high in use. Obviously, things like the nginx proxy as well. So we see a lot of that. And we see a lot of persistent storage. It still surprises me how many people are actually running NFS as their persistent storage connection mechanism of choice. Interface historically hasn't been seen as the fastest way to access storage, but it seems to be the easiest way of connecting external storage devices to Kubernetes. So therefore, it is seemingly becoming one of the most popular.

We also see a lot of people using Prometheus, and Grafana, and Alertmanager. There seems to be a relatively de facto standard now. So yeah, that is a very common deployment option. So you basically have that monitoring your application performance, setting up alerts, and that's actually why in the future we want Portainer to integrate with these tools as well. So you can have a single dashboard to see and engage and manage with all your applications. So using Portainer for certain native capabilities and Portainer integrating with things like Prometheus and Grafana to pull in metrics, so you can still just use Portainer as your single interface, but get access to best of breed capability from both Portainer and external tools.

[00:25:32] JM: Do you think the usage of service mesh or the widespread popularity of service mesh is overhyped?

[00:25:42] NC: I think service mesh is incredibly complicated. And I think it is still very early adopter-centric. So is it overhyped? Is it over marketed? Maybe? I'm not entirely sure. It's uses as widespread as many would be led to believe. It is still a very, very complicated deployment option. So if you want to get started with that, you better make sure that you've got some relatively senior people on your staff that can manage that level of complexity. So is it something that may take off in a far grander scale in the future? Sure, absolutely. But Kubernetes is still definitely not mainstream. So until Kubernetes gets more mainstream, then service mesh is still a long way behind.

[00:26:28] JM: What has been the go-to-market strategy like? Just because I know you must be competing so directly with all of these other container management solutions. I just like to know about the competitive dynamics.

[00:26:43] NC: Okay. So there's a couple of ways that you go-to-market, right? So there's the go-to-market through the open source product. And we have a very rich open source product, about 80% or even more of our dev efforts go into maintaining the open source product. So adding new features every single day. In fact, just last night, we pushed a new version of our open source product, and there were 19 new features added to it. So there's a lot of work in the open source product. And we want to get to the massive groundswell with the open source product. Get as many people using it as possible.

In regards to converting some of those to paid users, yep, we hope that as well. And we have certain features in our Portainer business version that are in this proprietary closed source version, things like advanced RBAC and governance and security capabilities that are really designed for the larger enterprise. And a very small subset of our open source users would likely convert to our commercial version. And that's kind of our goal. So we're not trying to say, “Here is the open source version. If you want support, pay for support.” Like some of the other open source vendors who convert, we're basically saying here is open source version of our product. It has about 80% to 85% of every feature you'll need. If you need some very specific controls around security or governance or compliance or chargeback or any other high-end enterprise functionality, they’re in our Portainer business product.

In regards to competing against others out there, I think we all have our strengths and weaknesses. So it's not very often that we compete head to head. Occasionally we do. And it's actually very rewarding for us seeing how we do against some incredibly well-funded competitors when we come out on top. So yeah, people like our simplicity. They like the fact that they don't have to retrain their staff to use Portainer. One of our very vocal users, he basically turned on Portainer for 600 of his staff on a Friday. On the Monday morning, they all came in, and were able to start using Portainer without any real training or anything. They just started using it. They were previously Windows people using .NET. Over the weekend, he transitioned their entire organization to .NET core on Linux. None of those staff needed any retraining on Linux or containers or anything. They just logged into Portainer on the Monday and started working. So it's that real simplicity, that real low barrier to entry that he find appealing from us. And that does not exist else out there in the ecosystem.

[00:29:30] JM: Is there anything else interesting you could say about the typical user of Portainer?

[00:29:37] NC: I would say the typical user is still to this day someone who has not yet bought in fully on things like GitOps or heavy automation. Beginning the journey into containers, they're beginning the journey towards automation. They can start to see the light and see the benefits of microservice-based applications and containers in general. They're still struggling to come in with the delta between more legacy virtual machine operating principles and the more modern container hyperdynamic principles and see Portainer as a learning tool to help them transition. That's why for us, it's really important to have the product grow and mature as the market grows and matures into things like full automation. But at this stage, I would say the vast majority of our users are not heavily invested in fully fledged CICD automation. They might have a CI pipe pipeline portal, but they don't have any real CD capabilities. So they have software to take their code and build images, but they still manually deploy and manage their applications.

That said, we still support most CD systems. So things like web hooks or Git integration. But moving forward, you want to get far, far tighter integration with things like Git operators, GitHub actions, so that we can have a whole bunch of automation triggering when our users mature sufficiently to want that degree of automation.

[00:31:09] JM: What's the hardest engineering problem you've had to solve so far?

[00:31:13] NC: When we decided to go into the edge compute space, Portainer was initially designed to be managing a dozen or so clusters concurrently. When we went into the edge compute space and we needed to now manage many thousands of clusters concurrently, how do you do that? We hadn't really thought of backend pagination. We hadn't really thought of monumental scale. So some of the some of the UX elements in the application that had dropped down boxes, when you click the drop down arrow and there're now 25,000 things in the list, that becomes crazy to scroll through. So we had to do a whole bunch of reengineering to say, “How do we have a UI that can support many, many thousands of clusters in a list and many, many thousands of containers that are there possibly?” So that was a huge undertaking. And we did that in partnership with Intel when we built the original edge compute capability. And that was a massive undertaking.

I suppose the second was when we did actually branch out from just being a Docker UI. I mean, if you know of Portainer’s background, we were just a simple Docker UI. And to make the transition from being a Docker UI into being a fully-fledged container management application, that was quite a monumental shift. So when we first introduced support for Kubernetes and ACI into Portainer, there was also a massive undertaking. It was basically a complete rewrite of Portainer from the ground up. So yeah, our version one product to our version two products, there is, I would say, maybe three times the amount of code in the version two product than there is in the version one. So pretty, pretty massive undertaking.

[00:32:54] JM: So you built out observability and monitoring into your platform? How do you build an observability product as kind of an add-on that competes with the companies that are entirely dedicated to observability?

[00:33:13] NC: Okay, so what we have done natively in the product is provided access to real-time stats. So real-time container stats, real-time host stats. We let you aggregate all the stats for all of the components that make up an application. So if an application comprises of a frontend, a middleware, and a backend container, we will let you aggregate all those together and see the stats for that. We can let you see the logs of the containers. What we don't do is let you see how was the performance yesterday at 10am versus today at 10am, or today versus

last month? We don't let you do that compare and contrast. We don't have any capacity forecasting at this point in time.

Now, we would be foolish to think that we could actually build something better than these other organizations out there. And that's why we are using integrations. So moving forward with Portainer is we want to start to bring even richer observability to let you do capacity analysis and looking backwards, looking forwards trending. We want to do integrations with things like Prometheus and Grafana. Prometheus as the data collection engine, Grafana as the dashboard tool, and then we'll pull those dashboards into Portainer and visualize them within the application. So we're not trying to replace the best of breed products, and we want to work closely with them through integrations.

[00:34:36] JM: And to what extent do you have to build out tools or API's for integrations? Like if somebody wants more robust observability or something, do you have to build out additional tooling for that? Or is it enough for them to just integrate with Docker or Kubernetes or whatever else, whatever infrastructure they have?

[00:34:59] NC: So we need to build out an array of dashboards. So we can actually build an integration with Prometheus and Grafana. And that's cool, but you get a blank dashboard. So in order for us to actually be able to get anything useful out of them, we need to publish our own dashboards. So we need to know what are the particular screens that we want to display within Portainer. We need to build those as dashboard templates, and then publish those into Grafana. So as part of the integrations, we have to build not only the API integration, but we need to build their portfolio or catalog of dashboards that we want to then publish through to then pull through into Portainer. So it's not as simple as saying take any existing backend Prometheus or Grafana installation, and Portainer magically can import all the configurations. No. We can reuse an existing implementation, but we need to publish our own dashboards into them so that we can then pull it out of back into Portainer to visualize in a unique way.

[00:35:59] JM: How much do you have to invest in support and help with the management of the platform? Is it self-serve enough that the average customer can totally take it on? Or do you feel like you need a lot of customer service on your team?

[00:36:14] NC: It's actually not that heavy to be frank. We've invested a lot in our documentation and in our YouTube tutorial videos. So we're aiming to have as much of it as self-service as possible. We have a small team of people whose sole job it is is to exist in the community and look at questions and provide answers. Now we do that because we want our users to have an amazing experience with Portainer regardless if its open source or commercial. So they exist in the community. And they answer their questions. They're in our Discord channels or in Slack. They're in StackOverflow, they're on Reddit. They exist everywhere. And they look and listen and answer questions as needed.

For our business customers, we have a full success program, which includes onboarding support and monthly check-in calls and ad hoc assistance to us. Surprisingly, even though it comes bundled with the business edition license fee, very few of our business users actually need to even use that service. The product is so turnkey stable, and the actual documentation for it is so extensive. Most of them are still relatively self-service. So even though they have really access to our engineering team, should they want to ask questions, or have demos or walkthroughs, or trainings, the vast majority of them don't find the need to do so because they've got sufficient traction in their deployment or their own management through our self-service tooling.

[00:37:47] JM: To what extent do you compete with the cloud provider offerings? Are their cloud provider offerings that you compete with? Or is it sort of agnostic because you're more like managing cloud resources?

[00:38:03] NC: I wouldn't say we compete with the cloud providers, again, because we're not trying to help people build clusters. We don't directly compete with cloud providers who want to just sell you a managed Kubernetes cluster. And in fact, we require you to have a cluster up and running. So having a user login to their cloud provider and say, "I want a managed Kubernetes cluster," is one of one of the first things they do when they deploy in Portainer.

There are some elements of cloud providers that, if you are all in on their technologies, may downplay some of the need for Portainer. If you're all in on, say, Google Cloud Run, that is a very simple tool to use. Do you necessarily need things like Portainer? Maybe. Maybe not. There are a lot of organizations though that are not all-in on any one cloud provider. They have

multiple cloud providers and want a tool that can normalize the engagement to the users, and Portainer can be that unifying portal. So users log into portainer. And it is the exact same experience, whether they're operating a Kubernetes cluster that's in Google, or Amazon, or Azure, or DigitalOcean, or any of the multitude of managed Kubernetes providers out there, that is a single unified experience between the user and Portainer.

[00:39:21] JM: As we begin to wrap up, I'd love to know your perspective on the future, like what you're planning to build out, and how you anticipate the ecosystem evolving over the near term.

[00:39:34] NC: So what we plan to build out is a serverless-like experience for our users. So for users who just want to deploy an application, they don't really care about where or how. They just want to deploy. We want to provide a neutral platform-independent interface to Portainer that they can just connect to **[inaudible 00:39:53]** serverless. So you connect to Portainer and you say, “I want to deploy my application. It's going to come from this Git repo or from this image. IT’s going to be published on the Internet. I don't need to persist any storage, or maybe I do. I want it to be auto scaled. Go.” And in Portainer, from that input, will then go and determine the best potential endpoint to go and run on based on metadata.

So the administrator will basically enrich a cluster with metadata that gives Portainer insight about its location. Is it a highly secure environment? Is it a non-prod environment? Is it a prod environment? And Portainer will digest that metadata and surface that to the user? And as part of that process of filling in this form or connecting to us through this API, we will ask them or have them provide us their answers to questions. What type of environment are you deploying? Is it prod? Is it a non-prod? Is this a high-secure or is it an insecure environment? We will use these as inputs coupled with the metadata to go and recommend an endpoint to deploy upon. We’ll then couple that with things like chargeback to say, “If you deploy this application on endpoint A, it'll cost you $15 a month. If you deploy this application on endpoint B, it'll cost you $27 a month.” W\e recommend endpoint A because it's cheaper. So we're going to have this kind of advanced metadata and near platform-independent interface.

We also want to get far tighter with things like GitHub with its full automation. So Portainer becomes more of a tool to monitor than deploy. So the deployments are all managed through

some external service that connects to Portainer. And in Portainer we'll go and connect to the backend infrastructure to go and deploy it. Again, you can maintain a library of multiple different types of GitOps connections to all of your different cloud providers, or you can maintain one to Portainer. And then Portainer will translate that for you on your behalf.

In the backend, we want Portainer to have more and more and more points of deployment relevance. So all we care about is that your application runs in a container. We don't care how and where that runs. It doesn't have to be in an orchestrator. So if you want to go and deploy that on Heroku's container platform, go ahead. If you want to deploy on fly.io, go ahead. So we want to add more and more and more deployment locations that we support in the backend. So in the frontend, user says, "I want to deploy my app in the backend." We have dozens and dozens of endpoint types that you can deploy to. So that's kind of the end goal for this year is to have this this neutral frontend, really rich array of backends we support. And then in the middle, we want Portainer having as many points of touch into an organization as possible. We don't just want the users of Portainer to be developers or administrators. We want users of Portainer also being the it executive or the finance person who needs to do chargeback. We want the person in security to be able to log into Portainer and say, "Show me how users are working and compliance against our policy. Have they actually configured network security policies? Is anyone out there running containers that are elevated and permissions?" So we want to have this multi-point of touch within an organization. So three pronged approach for the application moving forward.

[00:43:16] JM: Cool. Well, Neil, I appreciate you coming on the show. Is there anything else you'd like to add?

[00:43:21] NC: No. All I can say is watch the space. And now that we've got our series A funding, we're really doubling down on development and working to create a world-class tool designed to manage your applications. So yeah, definitely watch the space. We're going to be evolving very, very quickly.

[00:43:39] JM: Great. Well, thank you.

[00:43:41] NC: Thanks a lot.

[END]

