**EPISODE 562**

[INTRODUCTION]

**[0:00:00.3] JM:** Monitoring a Kubernetes cluster allows operators to track the resource utilization of the containers within that cluster. In today's episode, Ilan Rabinovitch joins the show to explore the different options for setting up monitoring and some common design patterns around Kubernetes logging and metrics gathering. Ilan is the VP of product and community at Datadog. Earlier in his career Ilan spent much of his time working with Linux and taking part in the Linux community. We discussed the similarities and differences between the evolution of Linux and that of Kubernetes.

In previous episodes, we've explored some common open source solutions for monitoring Kubernetes including Prometheus and the EFK stack, that's Elasticsearch, Fluentd, Kibana, and since Ilan works at Datadog, we explored how hosted solutions compared to self-managed monitoring. We also talked about how to assess different hosted solutions such as those from a large cloud provider like AWS versus vendors that are specifically focused on monitoring.

Full disclosure; Datadog is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

**[0:01:23.6] JM:** Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin prepares for these outages. Gremlin provides resilience as a service using Chaos engineering techniques pioneered at Netflix and Amazon. Prepare your team for disaster by proactively testing failure scenarios. Max out CPU, black hole or slow down network traffic to a dependency, terminate processes and hosts. Each of these shows how your system reacts allowing you to harden things before a production incident.

Check out Gremlin and get a free demo by going to gremlin.com/sedaily. That's gremlin.com/sedaily to get your demo of how gremlin can help you prepare with resilience as a service.

[INTERVIEW]

**[0:02:22.0] JM:** Ilan Rabinovitch, welcome to Software Engineering Daily.

**[0:02:24.4] IR:** Thanks for having me.

**[0:02:26.0] JM:** You've had a variety of operational roles in the past. You've spent at ton of time with Linux, and the main focus of the conversation today is going to be Kubernetes and specifically monitoring Kubernetes. But given your experience in the Linux community, how does the Kubernetes world and the growth of Kubernetes compared to those early Linux days?

**[0:02:49.8] IR:** I think containers in general and Kubernetes specifically, I've been seeing a ton of growth in the last couple of years. We publish a Docker adoption study every year at Datadog where we're just seeing it's, in some cases, 5 X'ing year-over-year growth on containers. So I think Kubernetes is sort of tied at the hip with containerization and so you see a ton of growth there. I think it took a while for the world to jump on the open source bandwagon were announced by — It's open by default and I think that's helped quite a bit.

I think the cloud ecosystem growing at the rate that it is has also helped quite a bit, but I would say containers and Kubernetes are probably — I would say are probably going at a faster clip than Linux was at this age.

**[0:03:30.7] JM:** That could be a byproduct of the technology or it could just be a byproduct of the volume of developers that have increased in that time span.

**[0:03:39.9] IR:** Yeah, I think it's a bit of both, right? The number of developers has increased significantly, of course. The number of folks that are using open-source platforms in general has increased as well. You used to meet some challenges and sometimes and sometimes bring you into certain organizations. These days, I don't think you can find any organization that's not using some kind of open-source. So that helps as well.

I think the speed at which folks are trying to develop and deploy to the cloud and just the sheer number of systems that we're all managing now has also forced us to take a look at new tooling and adopt it quickly when it helps us manage at scale.

**[0:04:15.1] JM:** You were a witness to the container orchestration wars. I was covering this topic pretty closely. You also saw the evolution around virtualization that happened before these wars. You saw AWS get started. You saw open stack. You probably saw VMWare, VMWare's rise. Do you have any broad takeaways from watching all of this virtualization technology come of age and quickly deprecate other pieces of technology in just all of a churn?

**[0:04:44.8] IR:** Yeah. I mean, I think the short story is everything old is new again, right? We were all time — Well, maybe I wasn't. It's before my days in technology, but at some point everybody was timesharing on large mainframes or shared computing platforms and we move towards more dedicated things. At the start of my — Towards the beginning of my career, VMWare started showing up everywhere as a way to save money. It will just run. Don't worry, we don't need 10 computers. We'll just run them all on one and it will be fine.

It tended to be an attempt at cost savings, although I think usually a poorly thought one, not that anything was bad with virtualization. Just, again, that idea that you're going to make, take the workloads of 10 machines and shove it onto like one equally sized machine and assume it's going to be the same is probably a mistake.

Over the years we saw Zen take sense [inaudible 0:05:32.9] that. Since then, containerization, technologies like OpenVZ started off. I'd say — Like I said, everything old is new again and things started to seem to run in circles. The goal always just seems to be the same; let's get work done faster and more efficiently weekly. I think it's pretty safe to assume that containers probably have an interesting — Will have some sort of a shelf life here as well and we'll see something new come along in the next 10 years that will supplant them or at least take some of the tension away again, or at least may be different than the containers we see today, m some evolution of them.

I would say the other thing I would — Open stack I would say is interesting as we're diving into — If we look at that specifically, was just that every industry seems to grow and shrink, and that open stack is exciting. I think though as more folks and more and more folks are focusing on public cloud or utility computing type situations, they're realizing they don't necessarily need to run — They don't necessarily have the expertise to run these types of things like — To run these

types of systems in-house. So they look at organizations like Amazon or Google or Azure, any other number of organizations out there. Where I see open stack focus these days is on the providers that need to run it, need to run it for their own customers or for very large organizations, whether it'd be telcos are large — Many, many tens of thousands of known environments. I don't know. We'll see. I'm curious to see what the dance between open stack and Kubernetes continues to look like.

**[0:06:59.2] JM:** Not to go down a rabbit hole, but open stack, that's — I think the ancestry there is that it was either pioneered or shepherded by Rackspace and some other companies that have a giant amount of infrastructure and the infrastructure is so tightly coupled with what they do that they would not move to the cloud in any time in the foreseeable.

**[0:07:22.4] IR:** They are the cloud.

**[0:07:23.2] JM:** They are a cloud. Right.

**[0:07:24.2] IR:** If you think about the providers that you're talking about there, they are the cloud.

**[0:07:29.9] JM:** Right, but open stack is for managing lower level infrastructure rather than — I mean, does it compete with Kubernetes or it has some sort of Venn diagram overlap with it?

**[0:07:42.0] IR:** I think there's a Venn diagram overlap. When I chat with folks, I hear sort of one or two stories. One is, well, of course, something needs to manage the underlying infrastructure, or IaaS, not infrastructure as a service provider, and I'm offering that up to my customers and they in turn run Kubernetes on top of that.

I think there was a use case in which some organizations may be were diving down that path, diving into open stack, diving into infrastructure management at a scale that they probably didn't need it, and in those situations it's possible that that abstraction layer may not necessarily be useful anymore. Working with Kubernetes directly on the host rather than offering everybody opportunities, spin up VM's on which they might in turn run Kubernetes. Maybe that meets their needs now.

**[0:08:24.0] JM:** What are the pros and cons of that model? I don't know if you're an expert in that, but the version of running Kubernetes on top of a bunch of VMs versus running Kubernetes just on bare-metal instances.

**[0:08:35.7] IR:** I mean, I think there's a bootstrapping problem that nobody likes to talk about, which is whether it's open stack or something else, you still got to get the software on the machine. You still got to provision them. Whether you're using some sort of pixie booting still or something else, you still got to get the bits onto the machine when it gets you from Dell or HP or one of the open compute vendors, right?

Whether it's for open stack or for Kubernetes, like that management layer is important. When I talk to some of our larger customers that may still be using — That are still running their environments in their own data center or have a very large data center of their own, they have lots of teams doing lots of different things. It's not just Kubernetes. So by offering up a common API on which you can provision VMs for your own needs, then that makes it simple for the infrastructure team to then in turn support a large swathe of folks. In terms of why you would want to run it directly, if you don't need a virtualization hit, why take a virtualization hit?

**[0:09:37.6] JM:** Yeah, fair enough. Kubernetes effect on competition between cloud providers has been exciting to watch, and in my coverage of this, initially I walked in thinking, "Okay. Kubernetes is a way for people to avoid lock in and be able to migrate from one cloud provider to another." That may be true, but it seems like in reality, what is happening is people are spinning up clusters of Kubernetes on multiple cloud providers. Are you seeing that same trend? Are you seeing people migrate from one cloud to another or are you seeing more likely people are just standing up Kubernetes on both clouds and using it as a kind of common runtime to interface with whatever cloud specific services they want on that particular cloud?

**[0:10:29.2] IR:** Yeah. When we talk to our customers, they fall into both of those camps, which is they — I guess or all three, which is they may be looking to move between one cloud provider and another and Kubernetes is offering them common way in which to think about their application deployments and their compute scheduling across those providers. There're others that are trying to do the multi cloud thing as you mentioned, which is great. Not everybody

definitely helps to mitigate some risk by being in multiple places, whether there'll be a negotiation risk or a technology risk or wanting to just be more geo-diverse in your deployment, even business requirements, sometimes you find yourself in situations where contractually you have to be in one provider or another.

But I think at the end of the day it's not about — It's really not about helping people move between cloud providers. It's about offering a simpler and more robust way to schedule your compute resources and the fact that it's letting folks move around between them, it's helpful, but I don't think that that's the primary purpose for most folks moving at Kubernetes?

If you're super invested in Amazon or you're super invested in your own data center, it's nice to have an out, but you've already built up a bunch of competency there. You're likely not just going to — It's likely not just like let me containerize all the things and we're going to move elsewhere, right? If you have a ton of data in S3, the data gravity of moving things out of S3 is quite high especially that the volumes that many of our customers have, right?

There's a great story in Wired from about a year ago from Dropbox about their move from Amazon to their own datacenters and the amount of effort and time it took them to orchestrate and just transfer all of the data that we all take for granted and stored in there in this nebulous cloud thing that they call Dropbox, out of S3 and out of EC2 and on to their own systems. That was a lot of work, and I don't have a ton of inside knowledge there. I'm sure there's a great show there for even the future about that whole project.

**[0:12:29.2] JM:** In the past actually.

**[0:12:30.0] IR:** Sorry. It turns out when there's a show every day, it's hard to keep up. Yeah. I mean, how much of that work is about like standardizing APIs and how much of it was just like the pipes only moved so fast?

**[0:12:41.3] JM:** Totally. Yeah, that was a great case study in infrastructure. We actually have a show airing this week with Tammy who was the lead of SRE, I think, on that project. SRE, you could imagine SRE on a project where you're building your own S3 like infrastructure had some interesting tales to tell.

**[0:13:04.5] IR:** Yeah. I'm sure that'll be fun. I know she's also doing quite a bit of interesting things with chaos engineering these days. I'm looking forward to that show.

**[0:13:10.9] JM:** That's right, and she talked about why to do chaos engineering even in the context of a massive data migration. It's promising for the idea of chaos engineering. If you're supposed to do it, you probably supposed to do it all the time. Let's get into monitoring. Today we're going to talk about how to observe and monitor Kubernetes, but to get there I think we should first talk about how people monitored raw Docker infrastructure, because Kubernetes has been around for a bit shorter period of time than Docker, and Kubernetes is managing Docker instances for most people. How did people monitor raw Docker infrastructure, or how do they if they have not moved into Kubernetes yet?

**[0:13:55.1] IR:** I think it depends on your environment. Some folks aren't using containers in the dynamic way that you might under Kubernetes or another scheduler, right? Just like folks had at some point were sort of lifting and shifting their work into the cloud. They're not necessarily thinking about these things may go away. It's just like, "I don't want to run VMWare anymore or I don't want to run my Zen Hosts anymore. Let me just go drop this on AWS. It will make it their problem."

Folks have taken — Early Docker users were, in some cases, doing the same whether it might be just using Ansible or some other tooling to just Docker run a container on their machine, in which case things don't change too much for you monitoring-wise, right? If it's static, the tools that you've had in the past likely still work for you. Things that you might want to get out of Docker while you're still there or the Docker socket offers a wealth of metrics via stats API that they offer there and you want to get that to be able to get the tales on all the different resources that your individuals containers are using, whether it'd be compute or memory resources or just any of the underlying raw resources.

The thing I'd say to everybody that's whether you're thinking about Docker in that sort of a static environment or a very dynamic environment like Kubernetes where we see containers churning all the time is don't forget to focus on your application. A lot of times when we start to dig in to monitoring, the first reaction is, "Well, let me look at all the resources on the machine. Let me

look at disc. Let me look at memory. Let me look at CPU. Let me look at how much throttling I'm seeing on this container," things like that. All of that is really important, but at the end of the day people are probably tired hearing me say this, but nobody ever calls you up in the middle of the night to say, "Hey, the website is running fine. All those orders, they're processing. We're good, but the CPU is a little high. Could I get a refund?" That just doesn't come up.

What you really want to focus on is your actual application, and that application could be the technology that you happen to be deploying there, whether it's a data storer or Redis or MySQL or PostgreS or something else and you want to just see — What you're interested in are things like the query times from those data storers and the throughput that they're doing and how are they performing for their end customers, or it could be your actual web application, in which case you want to look at your 500s and your 200s. Just make sure that your actual application is responding in a way that your customers are going to be happy with and are going to keep paying you, and those could be internal customers or external customers.

[SPONSOR MESSAGE]

**[0:16:26.3] JM:** This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM. Now, Datadog has application performance monitoring for Java.

Start monitoring your micro-services today with a free trial, and as a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineering daily.com/datadog. That's softwareengineeringdaily.com/datadog.

Thank you, Datadog.

[INTERVIEW CONTINUED]

**[0:17:20.5] JM:** Were there aspects of monitoring that were difficult in a scenario where there was no container orchestrator in the pre-Kubernetes world or people just had these Docker containers?

**[0:17:31.6] IR:** I mean, there's always the aspect of monitoring that became difficult with containers is very similar to what was difficult with VMs in the cloud when you were doing sort of auto-scaling groups. We've just sort of turned it up to, I guess, in non-orchestrated world. We turned it up maybe to five. Once you get to orchestration, you turn it up to 11 as this, and things are just moving around all the time.

I think there's — What folks started to realize as they moved there is that maybe some of their — Is that they really couldn't use monitoring that relied on static, static configuration files, monitoring tools that relied on static configuration files to track where things were like, it doesn't matter how tight your chevron loops are how quickly your converging if you got it — What? On a five minute converge loop? That's great. Whatever you were using to deploy your containers very quickly, whether it was static and the way you're discussing or Kubernetes later, right? It's moving faster than your monitoring system, and that's going to be a problem.

**[0:18:30.4] JM:** So engineering teams, they're moving to Kubernetes, and we've covered this in previous episodes. When people move on to Kubernetes, they're going to have some existing monitoring infrastructure in place. Once they add Kubernetes, are there new things that need to be monitored?

**[0:18:49.3] IR:** I think similar to the conversations we had earlier, there's all kinds of things that need to be monitored about Kubernetes itself, whether it'd be the control plane or the API server or the kublets on each of the nodes. So you want to be able to pull all that data in.

What becomes interesting is, again, trying to monitor your application as it moves around like that, and there's all kinds of different — As Kubernetes starts to move it around for you. There's all kinds of interesting approaches. Whether you're using Datadog and doing that or there's a couple of approaches where you can put some of your monitoring tooling in a sidecar next to your application on Kubernetes. There's pool-based approaches with things like Prometheus.

You want to make sure that your tools can team up with you as your workloads are shifting underneath your feet. That's by design, right?

In the static environment of our data centers of old where you started on your first day and they assigned you a server and you said like, "There are many like it, but this one's mine. I'll keep it running for my time here at —" whatever company you're at. It was okay to set up your monitoring and you know what normal was and it would alert you when something was different.

When normal is now the VM is changing every five minutes before of auto-scaling groups at your cloud providers and the container is changing even faster than that because of Kubernetes scheduling, or your other schedule are moving things on your behalf, you're going to want to — Normal changes all the time. So you're going to want to be able to figure out how to do that in some sort of programmatic or algorithmic way, figure out where things should be — To determine what your service health is and sort of monitor it from that perspective.

Being able to monitor — I guess the other things I'd look at in the world of Kubernetes are not just individual pods, but rather like your service as a whole or your deployment as a whole and seeing whether there are other enough replicas that — Do the replicas match what you expect it to schedule, things of that nature.

**[0:20:42.9] JM:** Talking about what we're going to monitor, you're always monitoring resource usage metrics, like CPU or memory usage. What are the other metrics that you're going to want to be looking at across your cluster?

**[0:20:59.4] IR:** I think you're going to want to look at those metrics that you mentioned and start to tell you to look at how your — If you're running the Kubernetes cluster, you're going to want to look at how efficient your users are taking advantage of of the resources that you're providing them. So you don't want to be able to look at that and break that down by application or by service or by pod, see kind of where your waste is.

In terms of the — There's other resource metrics of course, like storage and network, etc. I think if you're a Kubernetes cluster administrator, what you should be thinking about is the service that you provide your customers, which are likely internal customers if you're providing it at a

company and you're going to want to look at your work metrics there, which is what you do there as a — The service that you're providing your organization is the schedule of compute. How many failures are you seeing as folks try to schedule workloads? How long does it take between when you schedule — Between when somebody tries to execute a deployment and when it actually occurs and you're going to want to dig into that — Again, I probably sound like a bit of a broken record, but you really want to look at it from the experiences that your internal customers are having and those are the folks that are deploying their apps.
There's all kinds of ways to access those metrics. It might be — Again, there's a number of APIs on the control plane. The kublets themselves offer some APIs as well. Digging into logs is helpful as well, but you're going to want to look again at how successful are you at offering your customer, the developer trying to deploy an application on Kubernetes at meeting their SLAs you have with them.

**[0:22:30.0] JM:** What about etcd? Etcd is the tool that is managing the consensus of the state of your cluster, and if etcd was having some kind of issue, that would be problematic for your cluster. Do I need to monitor that or can I just trust etcd?

**[0:22:49.9] IR:** I mean, I think like all things. You definitely want to monitor that etcd. Not having that leaves you in sort of in a state that's not ideal. I mean, the cluster itself won't fail, but you're going to be in a spot where you might not be able to schedule or make changes there.

I think all the components that underlie Kubernetes, etcd is something that you're going to want to — You don't want to keep an eye on and we offer a way to integrate with etcd and pull all the metrics from there. I think another thing to keep in mind is that a lot of folks that are running etcd often may run it off cluster and separate from their Kubernetes infrastructure. If they're doing that, then you're likely not benefitting from the scheduling and sort of guarantees that Kubernetes might offer you for that etcd cluster.

**[0:23:29.5] JM:** Kubernetes has a base monitoring platform called Heapster, and Heapster aggregates monitoring and event data. It runs in its own pod and it's communicating with the nodes across the cluster by querying the kublets on those nodes. Can you talk about what Heapster does and why it exists in a Kubernetes cluster?

**[0:23:53.1] IR:** Sure. I mean, Heapster is sort of the original metrics aggregation and monitoring tool that came with Kubernetes by default. The frequency of collection is user configurable. I think it sets about a default by a minute, but you can get it down into the high seconds level range. Folks tend to use it — In the past I've used it, because it was there by default. These days, it's in the process of being deprecated, to my understanding from instrumentation sig. Folks are starting to focus more on some of the new metrics API.

Heapster at the moment supports a number of different syncs or backends where it flushes those metrics to. So that might be — There's ones like StatsD. There's Influx, there's Google Stack Driver and a bunch of other vendor solutions. So there was a lot of challenges with keeping up with those implementations and the differences between how those different syncs interacted and stored and propagated the data. The community has sort of moved into a different direction these days. That being said, it's still — In the past it was useful for things like digging into like how to do auto scaling and other things that required metrics within Kubernetes itself.

**[0:25:06.9] JM:** That term sig, that is special interest group. You mentioned that there is a instrumentation special interest group. This is a group where they discuss the state of monitoring and logging and instrumentation around Kubernetes?

**[0:25:20.3] IR:** Yeah. Kubernetes has a sig for pretty much everything. If there's more than two people in the world that want to do a thing, there's likely a sig for it. It's maybe a little bit of an exaggeration, but the instrumentation sig is where all the different monitoring vendors as well as the core Kubernetes developers that work on the metrics subsystems get together on a regular basis to chat about how to collect — About what those APIs should look like, what the future of things like Heapster looks like and sort of the roadmap for monitoring and metrics in the world of Kubernetes.

**[0:25:51.0] JM:** This is one thing I find interesting about the Kubernetes community, is that it is — There are all these commercial interests. You go to a KubeCon and there are so many vendors there. It does make me wonder, what's the diplomacy like there, or is there a very shared sensibility to what APIs Kubernetes should offer that make it easy for any kind of instrumentation company vendor to build on top of Kubernetes?

**[0:26:21.5] IR:** Yeah, it's surprisingly cordial. I mean, I think it's — And so far it feels better, I mean, feels better than what I saw in the world of open stack maybe 5+ years ago where in the world of open stack it felt like every vendor would show up and there would be some implementation — They'd be looking to create an API or a service that represented their technology solution in a way that made it work with open stack. It was always interesting to see what actually got accepted there and what moved in.

In the world of Kubernetes, it feels like there's better abstraction layers. I don't want to sound — I guess maybe that sounds a little negative on the open stack side, but it feels like the sigs have done a better job of ensuring that we come together and work towards that common standard that we all implement. So right now, you can see that instrumentation sig where there's been a bunch of new metrics APIs before it — Sorry. There's been a development of a new metrics API. As that metrics API matured, you saw a number of vendors come forward with sort of implementations of their own for how to consume [inaudible 0:27:24.1] metrics via —

**[0:27:25.7] JM:** Yeah. So those metrics on a cluster, do they get stored automatically or does it come down to the vendor choice or the metrics implementation choice to decide how to store metrics, how long you want to keep them around for how they can be queried. Is that the job of this kind of tooling that people plug into their Kubernetes cluster?

**[0:27:51.1] IR:** I mean, of course the Heapster bit that we're talking about earlier — Yeah, that was — I mean, by default it used InfluxDB, although there was a bunch of different choices you could make. A lot of the tales though that you mentioned around how things are stored and how long you're going to keep it around for and at what granularity are you going to roll them up, or not roll them up and how do you alert on them, what do you with them? That's sort of all belong in the world of that monitoring tooling and the vendors that consume from those APIs.

**[0:28:18.6] JM:** Where do the logs from a Kubernetes pod get written to?

**[0:28:23.1] IR:** It's interesting. Kubernetes doesn't really have a centralized logging offering. They get — Container users will generally log to standard error or standard out from their applications where that's a generalize pattern with the world of Docker as well. What folks tend

to do is run either a sidecar that will monitor that standard error or standard out and grab that or they run a daemon set, which I'm sure that they've got a container on each host collecting all the logs. They're not really centralized in any sort of a way in the world of Kubernetes.

**[0:28:55.7] JM:** Describe that sidecar pattern. How do people use the sidecar pattern for logging in Kubernetes?

**[0:29:00.2] IR:** In Kubernetes you have the concept of a pod which is a bunch of — Which is one or more containers that are going to be deployed together. What the sidecar pattern is, is taking a utility, whether it'd a monitoring or a log collection agent and you're including in that pod alongside your application workloads. There are other reasons you might want to include multiple containers in a pod that are actually more relevant to your application in terms of having multiple pieces that are actually working together to serve the traffic that that container may need to — Or the work that that container may need to do.

In the case of monitoring your metrics or logs, you're going to have that — Your collection agent alongside your application and picking up those logs and sending them off to your — Whether it's your internal storer, which is something like Elk, or if it's a hosted provider like Datadog, you're sort of looking to have a separation of concerns within your application and that those sidecars are taking care of it.

**[0:29:59.2] JM:** Yeah, I think there's also another popular stack, was it the F stack or something? We did a show about FluentD a while back. What is FluentD? What's the role that FluentD plays in a logging pipeline?

**[0:30:14.1] IR:** I mean, I'm not by no means a FluentD expert, but FluentD seems to be quite popular in the world of Kubernetes. It is one way in which you can — You sort of route your logs to the various solutions you may want. . It has ended up in the CNCF alongside Kubernetes and some of the other tools, but it's a data collector that you can use to route your logs to wherever you may want them. FluentD is one of the ingest solutions that Datadog would support, for example, for our own logging pipeline and product.

**[0:30:49.7] JM:** When we're talking about these kind of solutions, I do want to talk about Datadog, because you're a company that makes monitoring tools. What do you have to build to support Kubernetes users?

**[0:31:01.7] IR:** Of course, we need to integrate with all the different pieces that make up a Kubernetes cluster, whether it's Kubernetes itself and collecting metrics from a lower level cluster, at a cluster level, whether that's be scheduling events or logs from your masters in the various API servers. It could be — We want to make sure that we monitor things like all your ingress controllers and your etcd and all of the other tools that kind of come together to make up a functioning cluster for you.

At the same time though, we find that folks are often focused too much on that. We do build out integrations for all those things and we've supported them since before Kubernetes 1.0. But the place where Datadog spends a lot of our time is on making it easy for application developers to monitor their applications on top of Kubernetes. Whether that's [inaudible 0:31:51.8] traces from your applications in the open tracing sense and making sure that you're able to see where your requests are spending their time and how they're crossing service boundaries and host boundaries or logs in terms of making sure that we can collect all your application logs and to initiate those cluster logs and make sure that we've kind of document — Made easy to deploy patterns for that.

Finally, with things like auto discovery, we want to make sure that we can make it easy for you to automatically detect containers as they're spinning up and down and monitor them for what's inside them, not just sort of like, "Oh! Here's another Docker container. Here's CPU or memory or what have you," but really being able to monitor the things that are inside those containers.

Most recently, one of the things that we've added, we added late last year, was a live process view and a live container view in the Datadog product. If you think about things like ctop, which lets you see container stats across your hosts or htop or top that's letting you focus on processes. You realize that folks have all of these metrics at their disposal for the health of their cluster and then they occasionally still catching people SSH-ing places or running to their kube control commands to get data from their cluster. We realized that a lot of that really had to do with data recency and wanting to have like a real-time index of all your data. Rather than having

you run ctop across different hosts. What we built was  a centralized way for you to be able to look at it across the Datadog product and then be able to dive into each of those containers and even see what processes are running inside them and the resources that they're using. Those are just some of the examples of the things that we've built to make Kubernetes monitoring interesting and useful and real-time.

[SPONSOR MESSAGE]

**[0:33:37.5] JM:** Engineer chaos. Don't let chaos engineer you. When systems fail, you need to be in control. VictorOps is the incident management tool that you need. VictorOPs provides a comprehensive view of your system by integrating with the large number of monitoring, alerting and communication tools that your team already uses. It's your one stop shop for diagnosing an incident, escalating and alerting the proper engineers, collaborating during the firefight and resolving the problem. Through detailed notes, logs, run books and graphs, you can see the information you need in real time and also create detailed post incident reviews after the fact.

Head to victorops.com/sedaily. That's victorops.com/sedaily to see how VictorOps can help you. Go to victorops. com/sedaily. Be victorious with VictorOps.

[INTERVIEW CONTINUED]

**[0:34:49.7] JM:** I know that at least one model of deploying Kubernetes monitoring is through the daemon set, which a daemon set makes a particular pod run on every node in the cluster. Is the daemon set is the approach that you take when you have users that want to deploy monitoring across an entire Kubernetes cluster? Is that what they're doing when they deploy Datadog? For example,  they're deploying a daemon set?0

**[0:35:18.0] IR:** Yeah. That's sort of the get up and running real quick model and that's the one we generally recommend as you deploy a daemon set on your cluster and automatically you're not monitoring overall cluster as well as have the ability to monitor the applications that are being scheduled on top of it. The daemon set is quite useful for monitoring any systems management tool, but that's the way we approach monitoring metrics, monitoring from metrics logs or tracking perspective by default. It's the recommended path.

In some cases, customers may not have — They may not be the cluster administrator. They may be running in a hosted environment where they don't have the ability to run a daemon set, like if it's a multi-tenant environment or maybe [inaudible 0:35:58.8]. In those situations, the sidecar approach is definitely something we can support as well.

**[0:36:03.9] JM:** When you're building a monitoring product, do you want to enable people to auto scale their cluster in response to things that are detected in the monitoring data that is being created within the Datadog product for example, or do you want to just make a tool for observing and somebody that's on call can just look at — Do you want the cluster to be able to respond to data that is generated by that monitoring product?

**[0:36:37.4] IR:** I think you want the option for that. In general, Datadog likes to be a benevolent observer, right? WE want to be able to watch what's happening in your environment and help you better understand it and not necessarily take particular actions on your behalf. That being said, we have a lot of a number of customers that will do things like whether it be pipe our data back into auto scaling groups at the various cloud providers to get more control there based on application metrics or service metrics, and that's a popular pattern.

Similarly, in the world of Kubernetes, you have folks that are doing pods auto scaling where they want to scale up and down the number of pods based on workloads and schedule more resources there. So you definitely want to be able to inject those metrics back into the cluster in some way. That's something that the Datadog team is working on.

**[0:37:25.3] JM:** Datadog is a hosted service in contrast to people who self-manage their own monitoring tool chains, whether it's open source or not. How important is the hosted service aspect of it? Because you talk to other people who are building hosted services, oftentimes it's useful to have some auto scaling component or the uptime guarantees, for example. How does the usage of a provider, a host of provider compare to self-managed or open source monitoring solutions?

**[0:37:59.0] IR:** I'm a big fan of open source. I've been involved in the open source community for most of my career, and so I think it's lovely to see that we're finally getting some attention on

open source tooling in our industry around monitoring and systems management. There is a sort of a bit of dark ages there for a little bit a few years back, but it's good to see that things are coming together again.

That being said, much like any other open source versus hosted or proprietary solution, really, what you're making is a decision there based on core competencies for your organization, time available, whether you want some of the support options that are available. Of course, there's lots of organizations that self-support on those open source solutions. I think the piece around out sourcing a lot of this problem to somebody else whose challenge that is is important and something to keep in mind, right? I used to run monitoring systems myself internally at various organizations when I was running my operations stack. To do it well, it's not a trivial amount of time, and that's before you even get to the point of figuring out what you wanted to monitor. Just thinking about how — The time series data store in which you keep all of these things. How are you going to scale that out? How are you going to keep it online? How are you going to monitor it not to sort of — I guess that's a bit of a catch 22, but you want to do all the things.

There's a lot of effort that goes into making those monitoring systems highly available and useful to the end users in your organization, and that's where vendors like Datadog come in and we help make that a much easier experience for you and a much more integrated experience across the different toolsets that you have, whether it'd be tracing metrics and logs, being able to switch between them and sort of pivot on a log and see the metrics that are associated with it or vice versa. That's quite important and quite helpful to sort of reducing your time to addressing issues if you're having an incident or identifying issues before your customers notice it. So I think that's where hosted providers like Datadog can really help reduce your workload and let you go back to focusing on building your own applications and building the things that make your company a success.

**[0:40:04.0] JM:** What's on the roadmap for Kubernetes support? What are you working on at Datadog?

**[0:40:08.7] IR:** Right now we've been — Let's see, there's a couple of things we've been working on. We've got some new features we'll be announcing at KubeCon here in a couple of weeks around visualizing your containers. Definitely keep an eye out of for those. We're

continuing to improve our support for things like supporting the Prometheus format, which will be probably about the time this episode is out that will have been released. Then finally we talked a little bit about horizontal pod auto scaling and the idea of wanting to bring your metrics back into your cluster operations, and that's also something that you'll see coming out here fairly soon from Datadog.

**[0:40:40.7] JM:** What do you think about these newer ways of managing resources like? It's obviously serverless. Then there's the standalone container instances, like the Azure container instances or the AWS Fargate instances. Have you looked much at these different solutions for managing resources?

**[0:40:59.3] IR:** Yeah. I had to spend a lot of time with AWS team on Fargate as that was getting ready to roll out. We were one of their launch partners. I got to spend a bit of time with that. ACI has a similar approach and it's something that we're playing with quite a bit as well. It's interesting, we were already saying for a while now that the host is not what folks want to focus on in terms of their monitoring systems. You wanted to focus on the services that run on top of them. This just helps us drive that point home. That's another great place where that sidecar approach really falls in that we were talking about earlier, is how do you monitor — There's no host on which to run our agent. There's no host in which to run something that does log collection. You want to drop those inside your tasks.

I think these are all great solutions that are helping folks focus on — Again, focus on their applications and innovate more quickly as they're trying to deploy the services that their businesses are either depend on or that their customers depend on. More and more abstractions like this is great.

**[0:42:00.3] JM:** The difference between using a cloud provider product versus using a non-cloud provider vendor-specific product, that's a choice that a lot of people are making decisions between. For example, I know AWS has their own suite of monitoring tools and Datadog has their own suite of monitoring tools. How do you talk to people about, like potential customers about the differences between like choosing, "Okay. You're already on AWS." They're thinking maybe we should just go with the AWS monitoring tools. How do you convince them to go in a different direction?

**[0:42:40.5] IR:** I think the features and the functionality that we offer really end up speaking for themselves as well as the ability to look across cloud providers. Most of the cloud providers, AWS included, they have a fairly course granularity at which they store their metrics. They rolled up those metrics overtime quite aggressively so that after days or weeks you're no longer able to say what exactly happened at a given point in time, instead you're looking at on average of what happened over the course of an hour or a five minute window, which is —

**[0:43:09.5] JM:** It sounds like garbage collection, like a log garbage collection.

**[0:43:12.7] IR:** I mean it's sort of what you had in [inaudible 0:43:14.1] when we were doing like cacti or any of the other monitoring tooling back in the day, right? It's a way to reduce cost. It's a way to improve performance as you're querying the data and looking at it overtime. Datadog doesn't do that. We keep all of the data for 15 months at full granularity. If you sent us a metric now and we come back next year the same time, you'll still have that and you'll be able to say like how did Software Engineering Daily impact the traffic on my website? I can tell you looking at that year over year, month over month, right? That's important.

Also, we have deep integrations with all the different cloud providers. It's not just what's on your server. They're also pulling metrics from your storage providers and from like S3 or Azure blob storage or similar services at Google and other cloud providers. The ability to kind of look at that super deeply, connect it with your application data from our integrations from the agent. Most of those cloud providers are really focused on those underlying resource metrics. Their monitoring tools are really focused on telling you are they providing you with the resources that were scheduled for you and are you using them, right? Whereas Datadog is focused on helping you monitor the 360 picture for your environment. So we want to help you monitor that application server or that data store or that web server that's sitting on top of your instances. The Datadog helps — Is going to help you look across all of those things, and then as well as cross cloud provider. That's usually — By the end of that conversation, people tend to realize that we're the right solution for them.

**[0:44:47.7] JM:** All right. Just to close off, I know you've been looking at the usage data of people who are working with Kubernetes and obviously Datadog has access to lots of

anonymized data about how people are operating their clusters. What kinds of surprising trends are you seeing involved in Kubernetes?

**[0:45:10.0] IR:** I mean, one thing we saw — One of the trends that — We have a whole study up on the website and maybe something we can link to from the show notes. But things like the frequency at which containers churn between Kubernetes and GCS. We found that Kubernetes user tend to have much shorter lifetimes on their workloads than folks on ECS do. Some other surprising things we've had, the number of people running stores inside containers, whether it's on Kubernetes or not on Kubernetes is much higher than we were expecting. Like in the top containers you have technologies people are monitoring, you have things like Elasticsearch and Redis and MySQL, MongdoDB. These are all things that I did not expect to see as we were digging into the data for containers early on or even later with Kubernetes. Stateful services are definitely becoming a more popular thing to run there. I'm just surprised by how quickly people are betting into it.

**[0:46:09.1] JM:** It's surprises you because it's bold to put your central data store of record on Kubernetes.

**[0:46:15.7] IR:** To be honest, it's hard for me to tell you what people are doing in there. I can tell you what technologies they're running, right? But it's hard for me to say, is there a master database sitting on top of Kubernetes? I think people tend to think about containers and orchestrator workloads as place to put their stateless services. As we see people running stateful services there, at least at first it was surprising. Every time we run their report, I'm just kind of wondering who's going to come out in the lead in terms of the technology we see there. That's been — When nginx came in number one, I was not surprised at all. It's definitely like the web server of choice it seems within the world of containers.

When I saw Elasticsearch coming in number two on Kubernetes, I was like, "I'm a little surprised by that, but okay." As we started to get into things like PostgreS and MySQL and MongoDB, like I was also a little surprised. These days it's just — Again, not to make — It's not a popularity contest, but it's just interesting to see what comes out on top if only to see how the pattern and how people are using containers and orchestrators are changing.

Other stats like I talked about, I think it's been interesting to see how deployment patterns and best practices have emerged with containers and with orchestrators, right? I think if we went back to the early days of any programming language of your choice, let's say, we'll go with Ruby and you had people playing with dependencies and their gem files. They probably start it off there. Also not pinning to particular versions and when things blew up, starting to build other tools or starting to walk that down in their environments.

I think we're seeing similar trend in the world of Kubernetes. We see something like 75% to 80% of our customers are pulling tons of images that have the latest tag on them where it's just like whatever Kubernetes — Every time Kubernetes spins up a pod, it's going to pull for whatever the latest container is and pull that down. That's maybe living a little dangerously. I think only about 0% of our customers last time we ran this study, we're pinning their container images to like specific versions. It'd be interesting to see how that progresses overtime and if folks start to get burned by that and lockdown to specific versions.

Yeah, I'm looking forward to dig into more stats. So if any of the listeners have specific things, their specific questions around Kubernetes you're interested in, just drop me a tweet. I'd love to dig into them in their next study.

**[0:48:40.7] JM:** We'll certainly try to cover the evolving notion of stateful applications on Kubernetes. Ilan, thanks for coming on Software Engineering Daily. It's been really great talking to you.

**[0:48:50.6] IR:** Thanks. Thanks for having me on, and it's been fun.

[END OF INTERVIEW]

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]