EPISODE 1225

[INTRODUCTION]

[00:00:00] JM: Observability is a key feature of a well-architected application because building an observability system for a cloud application can be challenging especially at scale, many organizations elect to use third-party observability platforms rather than building internal tools. But these third-party providers often charge by volume of data collected which can be unpredictable and difficult to control. Organizations seeking to make their systems observable face to trade-off between convenience and control. Opstrace is building an observability platform that seeks to circumvent that trade-off by matching costs to the users who get value from their service. Opstrace's platform is open source and its clusters can be created easily via command line tools. Opstrace clusters talk directly to your cloud provider and can store data safely and inexpensively in S3 or GCS buckets.

Sébastien Pahl is the co-founder and CEO of Opstrace. Prior to founding Opstrace, he was a director of engineering at Red Hat and Mesosphere and he joins the show today to talk about the evolution of the market for cloud observable tools. How Opstrace is bringing observability costs under control and what the future has in store for the Opstrace open source project.

Sébastien, welcome to the show.

[00:01:10] SP: Thank you.

[00:01:12] JM: You work on Opstrace, and Opstrace is based around observability. It's an open source observability alternative to things like Datadog or SignalFX or other cloud hosted observability solutions. Why is an open source version of observability important? Why do we need an alternative to something like Datadog?

[00:01:36] SP: Yeah, thanks for asking that question. So our target users, the reason we built this project are the end users, the people who don't – There are a lot of great observability

solutions in open source. We use them. We build on them, but they require experts, and if you're not an expert you have two choices, you either use something like Datadog or SignalFX, a SaaS so to speak when you're at the lower end of the costs spend that you have. And then when you move over and you want to use these open source software solutions that exist today, they require experts to run them. They require people who know what they're doing or at least who are going to spend the time learning them and dedicating a good chunk of time learning how to do it.

So we decided to build something that is for the end user. What other companies are either building a SaaS or for their enterprise customers we decided to build in a way that can give it access to anyone, right? Anyone who just wants to hook up their Kubernetes clusters or non-Kubernetes clusters and start monitoring. So this is really to bring like the simplicity of things like GitLab to the world of monitoring.

[00:02:50] JM: Absolutely. And is there a particular class of user that wants this type of solution?

[00:02:56] SP: Yeah. So there are two types of users. Right now, if you want this, you can go and you can be – If you're in enterprise customers, you are pretty much well served. At least in the beginning, there's a lot of solutions out there that specialize for you because you have the money to spend. But if you don't have the money to spend and you already started sending a lot of data to somebody someplace like Datadog, let's say you're doing a million active metrics a month. There, already, using SaaS becomes quite expensive. It's about 10X cheaper if you do it yourself with open source, but only if everything is done for you. Not if you have to build it all and if you have to hire experts to do it. So the kind of people are people that are growing and that are having more and more costs in terms of logs, in terms of metrics that want to start to bring them in-house, right? Another example is also people who don't want to send their data outside of their own network. I'll touch on that probably a bit later.

[00:04:04] JM: Can you describe the architecture of an Opstrace cluster? How does it get deployed?

[00:04:12] SP: So we made the choice of automating everything from the deployment of the instances in your cloud accounts that Opstrace runs on to what's running there. So we start – The user gets a command line tool and the command line tool deploys a Kubernetes cluster. So we do heavily rely on Kubernetes. We use the cloud providers Kubernetes implementations like GKE and EKS on AWS. And so our tool spins up the entire infrastructure that is needed to run the observability platform. So we start by a Kubernetes cluster. Then on top of that Kubernetes cluster we deploy what we call a controller, an operator, which is basically – We're not using Helm charts to deploy this. Everything is automated. So our controller gets deployed. It's a piece of code. That's one of the parts that is completely open source and it runs and then it starts deploying open source APIs like Cortex or like Loki or like Prometheus itself, like Grafana, in ways that you can then start using. So it manages all of this by itself.

We also use a lot of other technologies. For example, one of the things that we do is out of the box. It creates TLS certificates and it creates authentication keys for the user. It's one of the big things that you don't usually get when you start with open source software, is while they usually are capable of security, they don't usually come with security out of the box. That's what we also deploy on top. So things, open source software like cert-manager. We hook it up to things like Let's Encrypt so that you automatically get TLS certificates. So our goal is not to reinvent the entire tool chain, but rather automate and bring it together for the end user. I'm happy to dive more into more technology parts like, for example, the entire controller and deployment is written in TypeScript. It's an interesting tidbit.

[00:06:11] JM: Sure. Let's go into the controller. So the idea is that when you utilize something like Kubernetes to deploy a stack like Opstrace, unfortunately it is thousands of lines of YAML, thousands of the lines of YAML that are of course generated by different tools, by different open source projects out there. Lots of things help you to do that. But then you can't just deploy these and hope for the best and then upgrade each individual component. So what we decided to do is code everything together. In this case, we chose TypeScript. We wrote code to deploy the infrastructure and then the code for the controller itself, which is anything that is deployed on top of the Kubernetes cluster itself. So if you go in and you kill a piece or you try

to change a piece of the infrastructure, the controller will put it back to where it was. It's also responsible for the upgrades of the system for eventually scaling it up or down when not needed. It's our way of completely automating the Kubernetes cluster.

We do choose Kubernetes, but that's just a choice that we chose to have an API to program against, which this controller is. Then for the end user, there's no need to know about the fact that it's running on a Kubernetes cluster. That's our hopes at least with building this controller. We're still early in our journey, but we still believe that automation is quite key to make open source easier than it is today, and the controller is part of that. So yeah, it handles everything from scaling the cluster, to managing its components.

[00:07:47] JM: Yeah. I think it would be useful if walk through the entire ingest path of some data being stored through Opstrace. So can you just go from the high-level to the low-level backend, how data travels through Opstrace?

[00:08:02] SP: Yeah, absolutely. So when an Opstrace cluster is running, the first thing that is exposed, you have a multi-tenant architecture. So thanks to Loki and Cortex, which are our databases and basically horizontally scalable Prometheus and log systems. Those are APIs that are exposed and that you can write logs or metrics in their respective format. In the case of Cortex, for example, it's the Prometheus remote write API. So with a Prometheus remote write API, you can hook up an existing Prometheus instance that you might have in your own cluster to set API and start sending the data there for long term storage and also querying. So the path starts in the customers, for example, Kubernetes cluster, but it also can be individual instances. So it starts at Prometheus. Prometheus collects data, collects the metrics and then starts sending these to the Cortex remote write API.

The way we expose this API is we have a little proxy in front of it inside of the Opstrace instance. So the proxy is responsible for authentication. So Cortex by itself is a multi-tenant system, but it's open by default. So we put an authenticated API in front of it. You need a token that gets generated during the install of the cluster so you configure your Prometheus with that token to authenticate to it and then the data flows through that proxy to Cortex. Then Cortex

itself is a fascinating piece of software because it will go through different phases there. Like Cortex has a piece called the distributors, which are responsible for deciding where to send what data. And then the most interesting piece is the ingester. The ingestor is a horizontally scalable piece of Cortex that allows you to store the time series data into a format that can be retrieved later. In our case, it uses a backend built with Thanos to store the data permanently in S3. So in the end, where the data lives, all the at the end is in S3. So you can think of an Opstrace cluster in a Cortex instance as big read and write caches for queries for metrics and for queries for – And for sending the data in, but the data ends up in S3. And when you query it, again, the data gets – If it's not in caches, gets fetched out of S3 and then shown to the user through regular Prometheus queries.

So the logs path is very similar. Again, use something like Fluentd or Promtail to send it to the Loki API. Loki then, which is a log-based system that works just like Cortex, stores these logs in S3 and then you can query them again. Why S3? That was one of the fundamental things that we wanted when we started Opstrace. We loved seeing companies like Snowflake and others utilizing the fact that you can actually store data in S3 quite efficiently. And then thankfully projects like Thanos and Cortex and Loki allow you to do it for time series and log-based data. And so it was a no-brainer to start using those as the backends. We're not in the business of reinventing databases. We're in the business of making them easy for the end-user. So yeah, that's the high-level, very quick path of data going in to the system and how it's stored.

[00:11:34] JM: I'd like to zoom out and put Opstrace in context. We've done a number of shows about M3, which is another open source system for scalable metrics and logging. And M3 I think was started around a scalable Prometheus solution. How does Opstrace compare to M3?

[00:12:00] SP: Yeah, a good question. From what I understand, when we looked into the different databases to choose, we ended up choosing Cortex for the reasons that it was the closest one at that time to Prometheus, but in a horizontally scalable way. And when we say horizontally scalable, we mean on the query and on the right path. But beyond being that, it

was basically exactly Prometheus, right? M3 then came out a bit later or at least I think it's been existing for a while in Uber, but like the open sourcing came in the last few years and they seem to be building a cool company around it. I don't have much comparison beyond. It seems to be a product to specialized as a database to store data. They might have more ambitions around it to build like user interfaces and more products around it as a company. But Opstrace itself is not a database, right? We specifically don't – We use existing database. We could have chosen M3. We decided to choose Cortex. But we are building a platform for the end user. That means we're building features that are there not only for managing it completely. And when we say completely, it's in a manner where the user doesn't have to understand how to scale it.

From my understanding, you still have to take M3 and either pay their company to help you with it or do it yourself, and that's fine. That's what a lot of open source databases and companies do. We're trying to work at the level above. The level above being the end user. So we're building tools around these time series databases. Like, for example, how do you go and scrape your existing cloud provider metrics easily writing functionality on top rather than focusing on the storage layer?

I honestly don't know if that's their focus. I don't want to speak ill of them. Like it's just not something that I've compared that much because, like I said, we're not building databases on our side does. That make sense to you?

[00:14:01] JM: Absolutely. And while we're getting broader context, I was just looking at your background, which is you've spent a lot of time in the modern cloud infrastructure space. Three years as a co-founder of Docker. Three years of Cloudflare. Three years at Mesosphere. You're at Red Hat. You basically spent 10 years, the last 10 years, watching the development of modern cloud infrastructure. Of all the opportunities that you could have started, why did you focus on Opstrace? And how does that fit into your thesis around where cloud infrastructure is going?

[00:14:42] SP: Yeah. So my co-founder, Matt, and I, we met at Mesosphere. That's where we worked together. And honestly when we started a company, we wanted to help in the infrastructure space in general just like you said. But the infrastructure space is vast. And after talking with a lot of people, we realized that while the observability technologies that we love are being widely adopted, there's a complete explosion of all the CNCF and CNCF-related type observability platforms coming out. They are, based on our research, of talking to people, right? Of doing a lot of interviews with companies. It was a problem that bubbled to the top, costs on one hand. We saw companies that were not monitoring their systems correctly. And this was not just by our research after deciding to start a company, but also remembering working with different big providers and seeing the state of their monitoring infrastructure.

So clearly something, like there was something missing because Datadog and others existed, right? Why, if they existed, were they not the solution to all of it? And if you dig into it you see that, yeah, people want to adopt these open source APIs. They loved of adopting things like Kubernetes going in the open ecosystem. You just got to continue making it easier for them. At least that's our hope. And whether it's cloud or monitoring, like mine and my co-founder's goal has always been making things easy for other people, right? Whether it was like – That's what I love what Cloudflare did as well, right? Like bring the CDN to anyone. Like everyone can use Cloudflare. Like that was a big inspiration for me. So that's another example, right? Everyone should be able to use containers. Everybody should be able to monitor their systems. Our system is not meant right away for everyone, but we'll start with the people who actually need the help and then we'll work our way up to bigger formats like enterprise.

So yeah, we chose that because after some research there was cool stuff to be built and lots of people to be helped especially in the way they monitor their infrastructure, right? Like if you think about it, all of this, this is infrastructure, we're talking about code and infrastructure, but in the end it wakes up people at night. Like it's sad that so many people are getting woken up by bad alerts and all of this, and like a lot can be helped by giving them better tools and better software. So yeah, we decided to choose that. It's exciting.

[00:17:21] JM: Tell me a little bit more about what elements of the Opstrace architecture are taken off-the-shelf and what you've built from scratch.

[00:17:31] SP: Yeah, that's a great question. So we currently built from scratch. So first things that are taking off-the-shelf, well, the first one I mentioned, Cortex and Loki are the two main – They are the databases we chose. When we looked at them, we looked, "Should we choose Elastic? Should we choose that? Should we choose –" We didn't look into M3 back then, but like there was Thanos versus going directly to Cortex. So we ended up choosing those two. Those are very important.

Grafana of course today is in the product as well. So what we've basically built is if you take Amazon today, you can go to amazon and you can pay them to get Cortex or Prometheus as a service and a Grafana as a service. You can think of our system as this, as a service Grafana or Prometheus as a service as well. It just happens to be open and works on multiple clouds, right? So we've built the installer we've built the controller that manages this lifetime of the system. We've built APIs to be able to ingest different types of data in there. For example, you can hook up a Datadog agent to Opstrace for migration or trying it out. Sending it there in parallel to Datadog and then it gets converted into Prometheus data. We've built that.

We're building a UI. That is something we're slowly building, but basically we're starting by doing UIs to help with building alerts, creating alerts, configuring alert manager, configuring the system, creating tenants, managing your keys. And then eventually we'll go further down the road and we'll build some alternatives to certain parts of Grafana as well because we want to show the data in different places. So what we've built is all these things on top and these glue to new APIs and automation. And then the last thing is none of this works if you don't test it correctly. So we have a we have a full open source integrated test suite of the system that runs it in two different clouds. So on AWS and on GCP we're also building upgrade tests. Because when you use open source software, what I meant with people need to be experts to understand how it works is it's all of that. It's upgrading your infrastructure, making sure that it works. So we're building and open sourcing these test suites so that you know that when

you're using something like Opstrace and you're moving from the version from last month to this month, we've tested it and we're showing you how we tested it so. So that's the thing.

And then off-the-shelf, I mean, so much – We're standing on these shoulder of giants in terms of off-the-shelf software, right? I mentioned Cortex and Loki, but you have to also mention Prometheus. You have to mention the Prometheus operator. I mentioned Grafana, cert manager, Kubernetes. Another very important one for DNS, which is external DNS – One thing that we give to everybody who wants to use Opstrace. One thing that's hard to use is while setting up a domain, sending up authentication. So we use Auth0 and we also have a DNS service to give Opstrace.io domains for people who are just using Opstrace because that's one of the things that you can't just easily go ask for a new domain. So these are things that we've built.

And so, yeah, we're standing on the shoulder of giants because you couldn't build something like Opstrace 10 years or even 5 years ago because the software to automate all this wasn't there. So, no, I love it. That's the same reason we chose to, by the way, make it an Apache 2 project. We have a few commercial features as well, but most of it is Apache 2 because we rely on Apache 2 software. So it wouldn't be fair to just create something that's not open out of other open things.

[00:21:25] JM: Tell me about how you go to market in an extremely crowded world when there are a million logging and monitoring tools and so many options for people to choose from.

[00:21:40] SP: Well, we are going in a place where there are not many choices, which is currently a whole suite of people who can't pay for enterprise and also can't – Enterprise meaning tailored deal, software to automate, of course, but also like it's expensive and also can't go to places like Datadog or SignalFX because they're already creating too much data, right? It's already costing them too much. They can't spend that amount of – They can't pay per bytes anymore. They can by doing it in a smart way, but not the margin of these places.

So the way we go to market here is we're building software that basically we're focused on building software that's going to work where we don't want to bullshit people. We want to make sure that they get what we promised them, and that's our go-to-market, is we're building something. We're proving that you can fully automate these solutions. We're taking our time to do it and then we're finding people who trust us. We find customers who are willing to say, “Hey. Oh! This might be early. Fine, but we're willing to like pay you to support our infrastructure because we believe in the vision that you're building. And the vision that we're building is that they're going to be able to pay us much less per month to have this fully managed than they would otherwise. So our approach really is first build the software. Make sure it works. Prove that it works, because very often like over the last decade how many people, Including myself in certain cases, have tried to sell the dream of things that are nicely automated and work out of the box? It requires a lot of discipline. We're funded enough to be able to focus on that discipline and build something that people that really only know how to hook up an agent to a platform like Datadog can in the end use and then will because it gives them a compatibility with everything else including eventually like other vendors. The market itself is so huge and growing. Like people need to monitor their software. Like our goal is to continue to – If we make good software, like I’m pretty sure there's enough people at least from what I see with our traction that will be just fine paying for this.

[00:24:12] JM: One of your selling points is that the cost effectiveness of Opstrace, you try to be more cost effective than a lot of the other services. Is there anything in particular that you've done around storage or network costs that you've managed to get the cost of ownership down?

[00:24:32] SP: Yeah. Basically it's not that big of a secret. So you take you take the data. You store it in a smart way the way Cortex and Loki are doing it in your own cloud account. And since it's your own cloud accounts, you get a couple of advantages. One, you're not paying costs to – Actually, you're not paying the network. For example, you're not paying AWS to send the data to another provider somewhere else. You get full control over how you store it. You get to pay the price that you negotiated with your own cloud provider. Like everybody who gets their certain size either negotiates. If somebody is small, they have credits. And since we don't

make margin on the amount of data that the person stores in their Opstrace system, but instead they pay Amazon or GCP directly, we're not incentivized to show them – We're incentivized to exactly show them how much it costs. We're going to be building into the system. How much it costs. Giving you the control like in limits to add more capacity, remove capacity. How long do you want to keep your data in S3? Keeping it for a long time is quite cheap and there's methods that we haven't even touched yet that can be used to store data for a long, long time, thanks to S3 cold storage.

But even before that, even today, just the fact that it runs in your accounts and that you're not paying somebody else a margin for the amount of data that you put in, it changes the game. Because since we're not storing the customers data, but the customer is, we can focus on selling them software, simply software, subscription to managed software. And we're incentivized to show them exactly how much it cost. We're incentivized to show them how to get the cost down. So by the beauty of storing it nicely in S3, like the SaaS providers do, you already get about – Like for metrics, like let's say for a million metrics, using something like Cortex the right way, you're paying already about 10X cheaper than Datadog for metrics. We even have a blog post for that. But why is it not 10X cheaper? Because you have to pay engineers to manage that software. That's what we're building. We're making it so that you don't have to pay engineers to manage that software.

[00:26:52] JM: What have been the hardest engineering problems that you've had to solve in building and deploying Opstrace?

[00:26:59] SP: Until now? Well, we started early with the different open source tools that we talked about, right? So starting early meant that we had to grow with them. We had to adapt to them. We had to learn them. So that was one thing. And then the other thing is making a system that already works on two clouds out of the box. It's easier to go from two to three or more clouds than from one to multiple clouds. So we spent a lot of time engineering our code base so that we can reliably stand up infrastructure on these cloud providers. And by reliably, we mean that most of the time it'll come up because having a small percentage of failure is okay when you're just bringing up one or a few clusters. But if you're bringing up thousands

like we eventually plan to for customers, you can't have constant failures. So the real challenge, the hardest part over the last time was test infrastructure and building good robust codes to set up the infra. We're far from done, but it requires discipline that companies often lack because it's easier at our stage, right? We're a smaller company. We can afford to focus on this right now. We don't have any other thing to do. So that's good, but it's hard code and it takes a while to design a system that will be able to run without your intervention. The good news is, again, standing on the shoulder of giants, right? The amount of things we learned at places like Red Hat, CoreOS, Mesosphere like help us with these things whether we learn through success or failure.

[00:28:47] JM: S are there any ways in which you feel behind the market leaders? So when you take a Datadog, they've been building software for such a long time. They have all these long tail integrations that they've already built out. Do you feel like you're at a deficiency because you can't cover the long tail of integrations that are needed to have a metrics platform?

[00:29:12] SP: Yeah. Let's be honest. We're very far behind the market leaders. That's the whole story, right? That's how you start. Like we're very, very far. What a place like Datadog has is that you can just use them, and you can just start using them and have it integrated with a lot of things, but it comes at a price, and that's okay. We can be far behind them. We're going to take our time to build the right amount of feature sets to get and satisfy customers. You don't need to have all the features, right? You need to be able to focus on the core problems that some people that you identify have with systems like Datadog. Datadog is not bad for everything. Datadog is an amazing company that built amazing stuff and really helped people start monitoring things without having to learn anything. We are just saying, “Oh! Look at that. Technology has evolved.” We can now take the technology that has been created over the last 10 years, right? And Datadog has been created I think 10, 12 years ago. I don't know exactly. But like companies like them have been doing that for that amount of time, but we can now take new technologies. And basically you take a few steps back and you rebuild a few things so that you eventually completely change how you do things and how you need to do things.

In our case, what is it to completely change how things are done? Completely change means the data always lives in your account. You pay your cloud provider instead of paying your metrics provider. That basically fundamentally changes. It compresses. Like it kind of compresses the pricing of these solutions, but who cares? The point is it brings technology to people who didn't have it before. Basically it's like we're building a mini Datadog and everybody can use, right? Whether they want to build a SaaS with it or whether they want to use it for themselves. Most people very quickly get to a scale. At least successful companies get to a scale where very quickly it's worth running it internally, but only if it's as easy to use like a Datadog. What does it mean as easy? It really means like all you do is like launch the thing and then hook up your system and then your users start logging in, right? So we're far behind that, and that's the goal. This entire thing is aspirational, but we know we can get there. That's the core premise. There's nothing in technology preventing us from building that vision. At least we don't think so. And so it's time to build it.

[00:31:57] JM: Take me inside the workflow and the day-to-day of a team that is using Opstrace or using modern observability tools in general. How is a modern observability stack fitting into the day-to-day workflow and like how is a Prometheus like fitting in in terms of instrumenting the stack to be observed and how is the data that is being stored and observed being used by that team?

[00:32:32] SP: Yeah. So one big entry point often is, at least in the case of the people that we're talking to and mine and the past of the team, what we're focusing on is really monitoring of infrastructure, right? Because this data can be used for a lot of things and is used for a lot of things inside of the companies. But if we focus on the monitoring and infrastructure, it starts with the developers and the infrastructure teams. Sometimes companies – Or very quickly companies get to a stage where there are people inside of the company that are focused on maintaining the infrastructure. They can be called SRE operations. There're many names and they can have various amount of technical knowledge, but they are usually the ones in charge of the infrastructure. And so they have to set up alerts for themselves or also help the teams inside of the company set up alerts so that they know what's happening and get woken up if something breaks. Alerts is, by the way, the end of the spectrum. It's like something went

wrong and you kind of already know that it could go wrong so you have something that will wake you up if it happens.

But beyond that, the data that gets collected, whether it's logs or metrics, it gets used every day for debugging, right? A customer writes in with a bug and somebody a developer from that SaaS platform will go on and we'll use either metrics. So it will do queries to the system to try to see what happened at that time. Maybe it's a recurring bug. So data gets used every day by, at the very least, operations and engineer, but most often nowadays thankfully also developers and others too, one, know if the system is still running. A bit like a plane, right? Like your plane has instruments and you got to know how it's doing. That's what allows you to fly the plane. Well, to manage the infrastructure, you have instruments. And whether you get woken up when there's a catastrophe, but also most of the time you use them to understand, “Do I need to add more capacity? Do I need to remove capacity?” And to do that, like when using open source, Datadog is a bit different, but when using open source, most people use something like Prometheus. That's the most popular system nowadays to collect the metrics to be able to query it and ask questions of these metrics. And then the only real game in town to query Prometheus is Grafana. So Grafana has integrations for many other things, but really shines at Prometheus. And there's lots of things that I think could be done better about Grafana, but it's discussions for another time, because itself, the great thing about it is that everybody in the company kind of uses it to query how things are going, right? Whether it's developers to understand and fix current or old bugs or whether it's infrastructure to know things like capacity. So I kind of went in all kinds of directions here answering the questions. So I’m happy to focus it more.

[00:35:52] JM: No. That's good. That's the benefit of asking an open-ended question. One thing I wanted to ask you about as a co-founder of Docker is Docker was this – This company that had extremely transformative technology, but relative to how transformative the technology was, it wasn't able to capture as much market share as people originally anticipated in the company and it seems like it was partially due to timing and just the way that the container orchestration wars played out, but do you have any reflections on your time at Docker and maybe strategic decisions that you wish the company would have done differently?

[00:36:39] SP: So I wasn't a Docker at that time. So I'm not able to like comment on the internals of how they decided to do things in general, but it's clear that I was at Cloudflare during that time and Mesosphere later. But it's clear that open source, like one of the problems that I think when you reflect back is Docker was wonderful, but it was very, very core to the infrastructure, right? It was very deep down in the infrastructure. And when you build something that's deep down in the infrastructure and everybody starts to rely on it, like everybody, for everything, whether it's development workflow, whether deploying to prod, right? It kind of exploded and started being used for everything. Then it starts rubbing more feathers with more APIs, more people, more things that people want to maybe re-implement, change. So it becomes complicated. And you really need to build – If you want to – I don't know this yet, the secrets of making money in open source, but I do believe that one of them is to build a complete big product. Not just a piece of infrastructure. You can see it also with things like Elastic, right? I think that Elastic is too core and that's why you see all this contention around it and you see people like – Because everybody uses something like the Elastic API.

So if you want to succeed, I think that the way to succeed in open source is to build complete products for the end user with user interfaces. GitLab is showing quite a success of doing that and for a good reason, right? Like build for the end user. Build something and use open source as a way to grow. Use open source as a way to give back. Uses an open source as a way to why should things be closed? Most of the code doesn't have to be closed, and it helps everyone if it's open, but it's a business model to choose to be open to be fair. And I think you need to treat it a bit more like GitLab is or a bit more like Red Hat is. Red Hat has succeeded in staying fully open source and nobody else has. But, again, when you see new commercial open source businesses coming, some are choosing to have more restrictive license. I'm a fan of the ones who choose to have completely open licenses and to go on the field of the product itself and of building something for the end user. And I think if you do that, there's enough companies that's out there that are willing to pay you whether it's a little bit per month for certain plans or even a lot per month for more complex help, for just making sure that their

infrastructure always works and doesn't go down, right? Yeah, go higher up the stack to make money in open source is I hope the way.

[00:39:34] JM: And is Kubernetes something that cloud providers are able to monetize effectively? Like are the hosted Kubernetes products getting a lot of traction in the cloud providers or do you know if people like deploy their own Kubernetes instances on top of raw virtual machines? Do you know what the most popular deployment mechanisms are?

[00:39:56] SP: So for those who do Kubernetes, what we at least in our questions or whether it's our own research or us and the people we know in the industry, the hosted Kubernetes – When people are using Kubernetes like us or others, they tend to go towards the hosted Kubernetes that the cloud providers are building. But don't take my word for it, because it's a tiny sample. What I do see a lot is a lot of non-Kubernetes. Kubernetes is not the only thing out there. It's very popular. We chose to automate on top of it because it has all these APIs we could automate, right? Like that's what it's about. Like we needed APIs to code against infrastructure. We needed APIs that were already baked in with the cloud provider infrastructure, right? Like when you scale up and down a Kubernetes cluster, if you use all of that right, like it does a lot for you, but it's not the end-all be-all. And I think it's definitely not easy, right? It's one of these things where, "Holy shit!" Like there's so much that you have to know to use Kubernetes, right? It looks easy in the beginning but that's a lie. So that's why I think a lot of people are – Like ECS is big. I mean, it's incredible to see how many people are deploying on ECS. And ECS is not the nicest thing to use, but it's easier. It's a bit easier. I don't know how Nomad is doing. Seems to be doing well, but I don't have any comments because I haven't seen it in the wild. But I'm glad that there's not only Kubernetes out there. That's cool. But I love Kubernetes itself. I'm not religious about it. It's an API. We can code against it. And we chose the cloud provider implementations. And I think a lot of people choose the cloud provider implementations because they do a lot of heavy lifting for you, but that's what I see, but small sample size.

[00:41:52] JM: Are you mostly building on AWS or Google Cloud?

[00:41:56] SP: Both. So from the get-go, we made sure that this works on both GCP and AWS. Why not Azure? Just because we didn't have time to do it, but we knew that doing at least two of them allows us to build the abstractions at least in the beginning a little bit so that it'll be easier to go to three. And that proved right because we had to rewrite couple of pieces many times and it definitely helped to build one single code base that does multiple cloud providers. Yeah, we use both. And by the way when we say we use those cloud providers, we only use Kubernetes and then the basics, the network, the object storage and then the Postgres databases like RDS, but just for a little bit of config. So we could even get rid of that eventually.

[00:42:51] JM: As we begin to wind down, I'd love to get your perspective on the broader industry as a whole. How do you expect the world of cloud infrastructure to change in the coming five to ten years?

[00:43:06] SP: I'm excited to see more and more of this software that actually – Like I said, it's complicated, but a lot of cool and interesting people are working to wire all these things together all over the world and make things easier. And you can see this. You can see younger startups come or teams and existing companies able to do so much more so much faster than before, and that's accelerating. It's a pain in the ass whether you're in the middle of it and you're building all the software. You feel like everything's broken, but that's the case for everybody in everybody's industry. When you zoom out, it's incredible how fast things are going and how much you can build now.

I mentioned, for example, Cloudflare, right? The fact that any person in the world can use it as a CDN for their stuff, and now like they're allowing you to deploy code on top of it, right? Like actual code. Run any type of code with their restrictions, but their worker system allow you to do that. So some people love to call it serverless. I don't care what it's called, but the idea is less and less worrying about instances, less and less worrying about like individual like how much RAM you need. I hope we're getting there. I think we're getting there like to really have this – Yeah, forget the machines a bit, because right now even running Opstrace, like that's what we're trying to abstract from people, right? Like stop seeing all these instances all the time and start seeing like pools of resources, APIs you can program against. Whether these

APIs are Kubernetes or workers or Lambdas, just think of the world as a big programmable thing, right? I think that's how people are slowly evolving towards. We'll see in another decade, but I'm super excited because you see more people working on things around the world than ever before. And I'm not just talking like the usual, like if the COVID crisis showed one thing, it's like the entire world is working remotely now and the amount of tech that's coming out of different countries is just fascinating. That's it. I'm just fascinated.

[00:45:19] JM: Well, Sébastien, thank you so much for coming on the show. It's been a real pleasure talking to you.

[00:45:24] SP: Thank you very much. It was a pleasure, a real delight.

[END]