EPISODE 1535

[INTERVIEW]

[0:00:01] LA: Hi. I'm Lee Atchison, and my guest today is Itiel Shwartz. Itiel is the Co-Founder and CTO of Komodor, a Kubernetes operation and management platform. Itiel, welcome, or should I say, welcome back to Software Engineering Daily.

[0:00:15] IS: Yeah, thank you. Happy to be here again and again with you, Lee. I'm super happy.

[0:00:21] LA: Great. Yeah, this is our first interview with video, so this is going to be a new experience for us. Hopefully, our listeners enjoy this as well.

[0:00:30] IS: Yeah, the listeners and our viewers, right?

[0:00:33] LA: And viewers. That's right. You're going to change my language here now. It's not just listeners. Well, we've talked before on this podcast, but recently, Komodor has made some changes in its direction. You've started focusing on a freemium and a dev first model. We definitely want to talk about those things. Can you start by telling our listeners who may not be familiar with you and what Komodor does? Can you tell me a little bit about what Komodor is? Who are you and what do you do?

[0:01:01] IS: Yeah, sure. I'd be happy to do a quick recap. What we do in life, like Komodor, we're a startup. We're still quite young, like a two-and-a-half-year-old. With that said, we are now 60 employees. We raised a couple of SC, then A and B round. We have dozens of different customers. We're a young startup, focused on helping people and we're going to talk about what people, to troubleshoot, cooperate, to manage Kubernetes cluster much more easily.

We started by focusing around the SRE and the DevOps people, and we do provide them really big value. But we are now doing a small focus change to help developers as well and not only operation people. That is with the mindset that we see in the ecosystem of the shift left movement, empowerment of developers. We see this as a very, very big trend, both for our

existing and potential customers. This is why we're doing a small adjustments in the product, to give real benefit for developers that find it hard to use to troubleshoot to manage Kubernetes. This is a bit about both history and current. Yeah.

[0:02:25] LA: Let's get right into the dev first transformation you're talking about. Obviously, the needs of an SRE and the needs of a dev are traditionally quite different. Now, DevOps is certainly merging them a lot more, but tell me a little bit about what does it mean to move from SRE-focused and operations only focused to be dev focused?

[0:02:52] IS: Yeah. Let me start by talking about the motivation, where like, why are we doing that? Why? Then I will talk about what really changed in terms of the product itself. We're doing the change, mainly because all of our customers ask us for that. We met customers, existing to potential customers. We usually met with the head of DevOps, head of SRE, heads of platform engineer, different names, same people, and so on. The number one request that we kept on getting is your product looks cool. It looks good. But can it also help my developers?

The first question we ask them is, yeah, we can. But why? Why should we have the developers? Why do you guys care? We're talking. Why do you want us to help the developers? The answer for that is a very, very, very large portion of the time being spent today in operation and in SRE force, is to give support for developers, basically. They're telling us that a huge chunk of their time is being spent on helping developers to understand what is happening inside their Kubernetes problems, and to solve their problems.

It's frustrating. It's a very large part of their job. Basically, what they are telling Komodor is, if you can I just take this very large chunk of time out of my plate, I'll be delightful. Even if you don't really bring me any value, and we do bring them value as well. Even if you don't bring me any value, any direct value, if you can empower the developers, if you can make them self-sufficient, I will be the happiest person ever.

We see this time after time after time and I can share with you like, yesterday we had a talk with one of, I would say, the biggest tech companies in the world, I would say. One of the top five, top 10 maybe. We talked with someone very senior, but hands on, head of all of the SRE, something, something. He has a big title. I asked him before. They were connected by a mutual

customer of us, who is a good friend of him. He already came quite happy and quite delightful to hear about Komodor. Before talking about the product, I asked him, how does the troubleshooting process looks like in your organization? Put Komodor aside, please help me to understand how does your day-to-day looks like.

Then he said that engineers, or SRE in his teams are spending 30%, 40% of the time answering tickets and Slack messages from the developers. He said, when a developer don't know what to do, it's very easy for them. All they need to do is to escalate it for us. All they need to do is write in the Slack channel, “Hey, my service is down. It looks like an ops problem. Can someone please take a look?” For the SRE –

[0:06:15] LA: There’s not being an ops problem, but ends up being something related to what they're responsible for.

[0:06:21] IS: Then he's like, “You know what, Itiel? 30% of the time, all they really need to do is read the log. That's it. They already have it in the application. Just read it. Why is it so hard for them?” We had a really interesting conversation. Now we're starting a POC with them. I think that what he said is the essence of a lot of the struggle in Kubernetes adoption, and in overall ops to dev relationship. I think that we are talking as a industry a lot about the shift left. I think, it happens. I think that in certain areas, like continuous delivery, it was something very ops back then.

Tools like Argo and GitOps are making it very dev. Even as a developer, I just don't think about it. I just commit to the master branch. A lot of magic happens and my code finds his way into production, which is quite amazing. On the other hand, once my code is in production, there is no set of tools, processes that help me manage the chaos, which is production systems. To fill these gaps, basically, this is what we do, this is what we focus. Empowerment and giving a self – we bring the developers into a self-sufficient state. We do that and now we ask like – Before I will explain how does it really translate into the product. I will stop for a second. What do you think, Lee? Does it make sense the story?

[0:08:06] LA: Well, it definitely makes sense. It also fits well into the move into much more DevOps focused. DevOps is a loaded word, but the idea that your developer owns more of

operations and the operations organization becomes more of the of the infrastructure and the supporting requirements to support a dev team to run their service within production. That's becoming more of a standard, more of a norm. I'm wondering here, if this shift left that you're talking about with your product is related to that transformation as well, too. Were your traditional customers more of the traditional dev and ops, and now you're shifting more into supporting customers that are more DevOps focused development teams? Then infrastructure focused operation teams. Is that a fair way to say?

[0:09:00] IS: I'm not sure. I think, because we are doing only Kubernetes. Only Kubernetes. This is our focus. This is our claim to fame and that's it. Even at the start, we were quite focused on cloud native companies, or companies that have a lot of workloads in the cloud, which are traditionally back then, like two years ago, it was more like the early adopters, the elite teams and so on. Not necessarily the companies where there's a very big separation. I will say that the biggest change that we are seeing is even those companies, it's like, the operation are doing a lot of things. There is deployment. There is secret management. There is so many different things which are like ops, or like DevOps.

I think that the day to operation, especially in Kubernetes is the last thing that is moving towards the developer, because it's the last thing in the list. First, you need to bundle the code, and then to test it and then to do a lot of other things. Only then it's in production. Only then there are problems, or only then you need to solve them. I think it's as an industry, it's not just us. I think, everyone has seen it, even a very cloud native companies. By the way, I think that tools such as backstage and platform engineers in general and platforms and so on, it's a very big buzz. I'm not sure when are people going to see this video.

Maybe in a year, it won't be the talk of the day, but I think those tools are gaining a lot of popularity, like a lot of usage, basically. It comes from the same reason. It comes with the same basically, reasons, or agenda of how can we make the dev much more efficient. We reached a state where we must bring them the appropriate tools. I think, Komodor is the tool that they need. But it's part of a bigger trend. I think that if I would say to you everything I'm saying now, like two years ago, I think almost no one would have listened to me. Basically, because we are still talking with companies. They are telling us, who can trust developers, like the operation

people? I don't want them to troubleshoot something that they are like stupid, they don't know how to do things.

There was one guy who said like, they're the last person in the company I trust on understanding production issues, or something like that. I think back then, it was like **[inaudible [0:11:47]** in general. It is shifting, but it's a small – it's like a gradual shift. It's not like, yeah, I support my developers, or I don't support my developers. A lot of the time, it's a question of giving them the proper tools to succeed. You can just tell them, “Yeah, how hard it is? You just open Splunk in New Relic and Argo and Kubectl and you understand what's the issue. I'm like, yeah it's four different tools that require expertise, required to understand what is happening, require deep Kubernetes knowledge. Komodor, by the way, don't come to replace those tools, or then like the Kubectl. We are replacing everything you need to do when it comes to interacting with the classroom.

We simplify everything for you. We take very, very busy and expert word and we simplify it. When it comes to the product, how did we change the product? We are trying to take screens that look very complex, or are really, really complex. We try to remove everything, which is not necessarily for the developer. We try to ask ourselves, I'm a developer, what I really care about? I don't really care about the internal events of Kubernetes during that time, or things like that, that might really interest the SRE. He just wants to know like, “Should I handle this issue? Is it an application problem? Is it an infra problem? If it's an application problem, do you mind showing me the logs as soon as possible?” For problem that just escalated as soon as possible.

[0:13:36] LA: The product changes that you're talking about, that make it more dev focused are more to simplify and streamline the displays to show only the information that the developer needs. How do you know what that is? Where do you draw the line, as far as developers don't need to know about this? Developers do need to know about this. Is it the same from customer to customer, or are those changes based on the maturity levels of the organization?

[0:14:05] IS: Yeah. Everyone is like a developer. There are different levels of experience between companies, inside the same company, inside the same team, right? There are different levels of experience. We're trying to answer in – the first thing that you will see when you have an issue in Komodor, we should provide meaningful value to around 90% of the people who are

going to look at the screen. Our goal is obviously, 100%. Sometimes we miss. We gave too much information, or we miss something.

I think that the canonical questions of solving an issue are quite similar, both for experts and non-experts. The expert can answer those questions much faster. What's the current status? How many replicas do I have? If I have zero, it's very bad. If I have nine out of 10, okay, maybe it's fine. What's the current status? How long do I experience the problem that I'm currently experiencing? Is it just started? Or is it something on? How frequent is this issue? Is it something that always happen? Or it started now for the first time after a couple of weeks? Was there a written change in the system? Was there a deployment, a config change, a vol change? Anything else? Is there any, either alerts ongoing in the system at the moment that might explain this? Is there a Kubernetes info problem, like node problem, or pressure that might explain the issue?

I ask a couple of questions here. I hope it wasn't too much, or too fast. I think, both the expert and the non-expert hear about these kinds of questions, the triage phase. An expert can answer those in one minute. I don't know, maybe even, like less five seconds. A developer might never know how to answer those questions. We try to take the most important questions and to put it in your face. A lot of the times, even inside Komodor, I see a graph and I tell them, we have a problem. This is not how a graph should look like. That's it. I don't need anything else.

Give me a memory graph, or I don't know. We just had a problem with how many requests per second, and I thought, I'm like, this graph doesn't make sense. Someone is lying. Or maybe, we're sending the wrong metrics. Or maybe the system is not behaving how it should behave. I saw a lot of sending that graph. This is not good. I intended that, because I know, I'm quite good in what I do, as a developer, CTO, architecture. I've been there and I know that. For me, looking at this graph, it just doesn't make sense. It doesn't tell the complete story.

Someone else might look at this graph and say, "I don't know. It's a graph. It's a nice graph. Is there a problem? How should I know?" Komodor try to take this knowledge that exist only for expert Kubernetes people, and to dumb it down, basically, for them. Telling them, "Hey, you are missing a secret. If you create a secret with this key, your application will be good. If not, it's not going to load. There is no middle ground here. You are missing a very key part." The PVC is not

loading. Why it is not loading? You will misconfigure it. You missed, did something. There are a lot of different failure scenarios. The gist of it is something that we believe we can really simplify. We already see that the first reaction from customers, and they are happy customers.

[0:18:04] LA: Let's go through a specific example. I want to focus specifically on the types of information that a traditional ops person would want to see in Komodor, and types of information that traditional dev would want to see in Komodor. Let's take a problem. Like nodes are constantly failing. The failure rate of nodes is 10%. 10% nodes fail every hour. Some astronomically high number. Something obviously is going wrong in this system that's causing that to happen.

Now, in order to diagnose that, that problem could be something in the infrastructure, it could be resource allocation issues, not enough – the clusters overloaded. It could be something like that, or it could be a code related issue, where services are starting up, they run for a period of time, they have a memory leak, they get shut off, because memory is being used up, and then that node goes down. Whatever. It could be a couple of different reasons. In one case, that's more of a ops focus thing. The Kubernetes cluster issues. The other one is very much a developer-focused thing. A problem in the code that's running within the node. How do you give information that's available to both of them to help diagnose what that problem is, and so they can determine on their own, yes, this is my problem, or no, this is not my problem? The other team gets the opposite answer, right? How do you do that?

[0:19:36] IS: Yeah, great question and a great example. Nodes, for example, is something common. A lot of customers complain, the nodes are going up, they are not – People are always having problem with nodes.

[0:19:48] LA: Common Kubernetes problem.

[0:19:49] IS: Network. It is quite common. From a dev perspective, usually, like 90% of the time, like 95% of the time, if I can simply tell him, “Hey, the cluster is having nodes issue,” he's happy, because he doesn't need to handle it. In most cases, he's going to escalate it for the developers.

[0:20:09] LA: But it could be a code startup that is causing this.

[0:20:12] IS: It can. No, it can. It can. For the dev, currently, only doing the triaging of understanding, like why did the dev came to Komodor? He's not bored. He came to Komodor, because now he has a problem with his application that will always bridge. There was a pager duty alert, someone complained that the system is not working. The developer now start doing the diagnostic, the triaging phase.

Without Komodor, he's reading the logs, going to sentry, maybe doing a lot of different things, not understanding what's the problem. If I can tell him, “Hey, your nodes are having issues,” in 95% of the time, it means, first of all, let's escalate it for the DevOps. Maybe now, we need to work side by side, like both developers and an operation person working together. In most companies, when they hear there is a node problem, the developer is not responsible on solving those issues.

I'm telling you right now, we can argue, we should be responsible for that. By the way, in a conversation like yesterday that they had, that they just mentioned, he said, developers don't understand sometimes that it's like a node problem. They are spending hours diagnosing something that they simply can solve. He blamed GKE, by the way. He said, GKE have a lot of node problems.

For the DevOps that comes to Komodor, now come the DevOps, he heard there are node problems, but why? We give him, first of all, a full timeline of everything that happened in the cluster, from the node point of view. All of the nodes that existed, why they died, what are their status, their error code, the logs, if there were any. We bring all of this data and we have a screen called node, and we have node issue detector. They can go there to understand and to find the issue. When they look at the breakdown of who use Komodor or for what scenarios, you see that the people who click on the node tab in Komodor when it comes to dev versus DevOps, it's 95% of people who are ops people, 5%, which are developer.

It's like, they don't really care. They don't care on one hand, but on the other hand, this caused them a lot of pain. Because when they get the page that your application is not working, they don't know that it's not their fault. A lot of the times, what happens is it was one time, it was an

old issue. Second time, it was an old issue. The third time that he gets the page, the first thing that he's doing, the developer, is to escalate it to the DevOps, without checking for anything, because it's much easier. You're writing like, "Hey, can you help me?"

We try to give him the tool to determine, what's the status? Is it an infra? Is it not an infra? Are there other pods in the same node also experiencing the same symptoms? Are they all suppressing? If so, again, it's like something bigger than one single developer. If all of the node is crashing all the time, it might be that one developer, they have done a change that caused all of these, or a huge spike in traffic, or things like that. There are countless of failure scenarios, but we help them doing the triage, and to try and find out the real root cause of it very, very fast. Again, fast and streamlined, you said it yourself. Yeah.

[0:23:47] LA: Cool. Cool. Let's talk a little bit more about the actual mechanics of this now. This DevOp, dev first focus, you've also changed the model for how you sell your product, right? You've moved to a freemium model. You're moving to a POD strategy. Can you talk a little bit about that, and why that change?

[0:24:13] IS: Yeah, sure. Yeah, we are moving. Now we have a freemium offering. It's in the market for only a couple of weeks, already thousands of users. It is kicking off quite nicely. The reason for that is again, like everything we do is we hear what people are asking us. We have customers, like customers who are now paying customers telling us, "Why didn't you have a freemium product? I would have tried that ages ago, but I simply couldn't." Because now we're targeting not only the upper ops guy or girl, but also the developer. Developers are much more used to freemium products, compared to sometimes operation people.

I've been in operation, sometimes book at them, or is the only way. It sucks, but that's life. Komodor also had 14 days free trial even prior to that. We are adding a lot of capabilities in the product that we want them to serve them as, basically. We did changes in the product that allows us to be the best Kubernetes dashboard in the market to operate, to manage. You can do everything you can do with Kubectl, but much easier with Komodor and much simpler with Komodor. We felt that in order to really gain the – first of all, to give the users something that will be valuable for them and for us to grow much faster as a business, that's the best solution there is.

We also released an open source as well. We took a chunk of the product. We simply open source it. It is called the helm dashboard. It's in the market for three months, I think. Like, ever since KubCon, basically. October, I think. I think so. Yeah, two and a half months. Which also gained a huge amount of popularity. Helm dashboard is 3K stars project, 100-plus daily users. It is much better than expected when it comes to, you know, I'm happy. We have almost 20 contributors already. I saw open-source project. Even I was quite surprised by the success of it.

Helm dashboard in its core is nothing too fancy. It's to take something like Helm, which is used daily by developers and operations people. To give them the proper UI to simplify it. I used Helm for six years, seven years. I don't know. A lot of time ever since, I think before Helm 1.0. I love Helm. I think, it's like, it changed the Kubernetes ecoystem. It did a lot of amazing things, Helm together with CRD. It's not that easy. It's not that easy. I really love that product, but it's not that easy. We gave, people who use Helm a very simple, or easier way of working with Helm, instead of the CLI. This in turn, really helped them enjoy life more.

We see a lot of people are telling us, "This really changed how I operate with Helm and this is exactly the reason we took this part of the product and when we moved to open source." But it's really fun to hear. It is really, again, you see that there are so many moving pieces when it comes to Kubernetes. There are so many gotchas, so many small things that you need to understand. When we first published the product and I sat down with the marketing team in Komodor, and we really have a really good PR. A lot of people shared it and so on. The beginning, they told me like, "Itiel, Helm looks like quite successful. Are you sure that it needs a UI? There's a CLI developers and ops people love CLI."

[0:28:36] LA: They're like dashboards are, too.

[0:28:38] IS: Yeah. When you see it, once we build the first mock up, you're like, wow. I can't believe I worked without that. I think that we're sometimes as developers, as operation people, we're used to working quite hard. Doing five grabs. To me, it looks like it makes total sense to get the data you need. Grab, auc, grab, auc, do anything to a file is in iPython. For me, sometimes it takes only one minute, or something like that. It works. For people who are not experts, having those tools is life changers. We are taking them from a state, where they had no

idea what is happening to a state where they can operate it and to own it and to feel accountable for that, without them needing to be a bash experts, or to install a lot of plugins, and so on.

[0:29:39] LA: The non-Kubernetes experts, such as the development group, this is a perfect model for them, because it gives them access to the information that they need about how Kubernetes is operating, and all the things that Helm provides, without having to learn the API, without having to learn the command line, I should say.

[0:29:58] IS: Yeah. Even very –

[0:30:00] LA: I know developers hate Kub control. I mean, they hate Kub control. They hate dealing with that.

[0:30:05] IS: I will say Kub control, 10X better than Helm CLI, I think. When it comes, do you use again and again in memory, like a lens, like I really the project. Every trivial things, like sometimes it doesn't show you pending Helm status, or something like that, when you do Helm LS. Like, “Ah, yeah. We don't show that.” You need to ask for this explicitly, or something like that, like small things that I'm like, “But why? But why? It's super valuable information. Why are you hiding this information from me?”

We are getting a lot of issues, by the way, good issues. Add this functionality. Add this functionality. Every time even, we are surprised on the gachas that happens. For now, as an example, we just got an issue that 10 dashboard doesn't support neatly cluster with more than 300 Helm charts installed. I was surprised. I didn't knew it's like a combination. We get like, “Yeah, support this, support this.” Yeah. It's like, we must have it. You see how many different failure scenarios, or good scenarios exists inside our ecosystem.

Komodor in general, when you think about Kubernetes, you should, first of all, try Komodor, so you will gain this level of ownership autonomy. Even having fun, right? I love Kubernetes. Enjoying a bit the work, not only living in fear.

[0:31:37] LA: Well, we're actually at time here now. This time went by really, really, really fast. It's been great talking to you. Is there anything else you want to mention about your new focus that you haven't talked about, do you think is important for our listeners, or watchers?

[0:31:56] IS: Not really. The only thing that I have to say is that we have a lot of surprise in store. You can follow us on Twitter, on LinkedIn. We are planning a couple of integration with very common tools in the ecosystem. I mentioned backstage, Argo CD. Our goal is to be a very integrated product with the community. We are also working on something still in early beta, but a desktop app for Komodor. Really, feel free to follow us, follow OD, which is our dev rail to any collab dates, or memes about Kubernetes. It's worth it for the –

[0:32:39] LA: Sounds great.

[0:32:40] IS: Yeah.

[0:32:41] LA: Great. If people want to learn more about Komodor, where do they go?

[0:32:47] IS: Komodor.com, Komodor Twitter, or to follow me on social.

[0:32:53] LA: Komodor is K-O-M-O-D-O-R.

[0:32:57] IS: Yeah. Sorry.

[0:33:00] LA: Yeah. Great, thank you. My guest today has Itiel Shwartz, who's the CTO and Co-Founder of Komodor. Itiel, it's great as always talking to you, and looking forward to talking to you again soon.

[0:33:13] IS: Yeah, it was a pleasure, Lee. Bye-bye.

[END]