EPISODE 1497

[INTRODUCTION]

[00:00:00] ANNOUNCER: This episode is hosted by Lee Atchison. Lee Atchison is a software architect, author and thought leader on cloud computing and application modernization. His most recent book, *Architecting for Scale* is an essential resource for technical teams looking to maintain high availability and manage risk in their cloud environments. Lee is the host of his podcast, Modern Digital Business, an engaging and informative podcast produced for people looking to build and grow their digital business, with the help of modern applications and processes developed for today's fast-moving business environment. Subscribe at mdb.fm and follow Lee at leeatchison.com.

[EPISODE]

[00:00:45] LA: Cloud native applications utilizing microservice architectures has grown into one of the most popular application architectural patterns in recent years. The value of leveraging dynamic cloud resources, along with the flexibility and scalability of microservice architectures creates a strong paradigm that's hard to miss. The strong adoption of Kubernetes has strengthened the pattern enormously.

The unique structure and requirements of Kubernetes has led to an increased need for Kubernetes specific monitoring and diagnostics tools. There has been a large number of companies that have jumped at this opportunity. One of those companies is Komodor, a Kubernetes diagnostics platform that focuses on Kubernetes troubleshooting for the entire Kubernetes stack. Itiel Shwartz is the Co-Founder and CTO of Komodor, and he joins us today.

[INTERVIEW]

[00:01:38] LA: Hello, Itiel. Welcome to Software Engineering Daily.

[00:01:42] IS: Hey, happy to be here. Super excited.

[00:01:45] LA: Thank you. Kubernetes is central to operating in modern microservice architecture applications. In your experience, where do most microservice related failures come from?

[00:02:00] IS: Oh, it's a really tricky question. I'm not sure if there's one very strong dominator in terms of who is most common. I think that all failures are different in their own way. There are a lot of different failure scenarios. I think that because of the complexity of microservices by itself, and the complexity of Kubernetes as a system, the combination of the two is that the two are greater than the sum of their parts. We see a lot, a lot of difficulties, both managing and understanding and solving microservice issues, or Kubernetes and facilities issue in general.

[00:02:45] LA: I guess, the real question I'm asking is, does it tended that the application itself that has problems, or does it tend to be the service architecture that has problems, or the Kubernetes infrastructure that has problems? Or is it a combination?

[00:03:02] IS: Yeah. I think it's a combination. Kubernetes by itself, mainly in the cloud provider is quite stable. Kubernetes does have issues in **[inaudible 00:03:11]**. They do have problems. Overall, they tend to be quite stable, which is good. The underlying infrastructure is not the main issue, or the main problem from my experience. It is somewhere in between the combination between the application and the infrastructure. It might be because you don't have enough **[inaudible 00:03:33]**, or it might be because you just configure the PVC, or the load balancer on, or the ingress itself.

There are lots of different places when things can go wrong. To go along when it comes to microservices in Kubernetes. I think that because the complexity of so many moving pieces, there isn't one golden place stakes, or to try and overcome, because the system complexity is usually, because of the people in the organization, the different needs of the architecture, and the customer needs and so on. The system just represent everything is happy inside the organization. I think, because the board is so comforting, and organization are so complex, that the system by itself is also very complex and Kubernetes just adds another flavor of complexity.

[00:04:32] LA: It's another layer of complexity on top of everything. Yeah, that makes sense. Yeah. It's not one cause, but it's a layer of complexity that's added on top of other layers of complexity. That's where –

[00:04:43] IS: Yeah. There's the knife meme. I think, the eight layers of Kubernetes published within an iceberg. The cool thing about Kubernetes is even when you like to master it, you find out that there's a lot of new things that the community release, or Kubernetes just released. Always, we need to stay up to date with the core infrastructure.

[00:05:09] LA: That makes sense. Talk about Kubernetes monitoring them. Obviously, monitoring the Kubernetes stack itself is important, but what are the key requirements that you need for monitoring a Kubernetes stack? What's needed to be a good quality monitoring system for Kubernetes?

[00:05:28] IS: Great question. I will give a full disclosure that Komodor is Kubernetes published within the system. When you think about most product opinion monitoring, you need to understand that there are the basic things that you must monitor, and then because we talk about all the layers of complexity, there are also different things that you should have in place, specifically for commands.

The basic, like what is called the three pillars of observability are obviously metrics, and logs and tracing. I will say that without metrics, will be lost in Kubernetes. You need to know the CPU, and the memory, these expression and so on for each one of the nodes and also, for each one of the application running on top of the node. I think Prometheus is an amazing tool with Grafana. It's not you have the same solution on relic and so on.

When it comes to logging, again, it's the basic today to research as **[inaudible 00:06:28]**, or Lock Key, or stump also give a really good visibility. When it comes to pricing and with the pricing, it looks like **[inaudible 00:06:39]** and also, data though, I think that can really help you understand and solve issues, especially issues in a microservice environment on one hand. On the other hand, I will say that, from my experience, most developers and most operation, people don't really know how to use APM, how to read the tracing, not one to understand. I think the tool by itself is super valuable. On the other hand, I think you acquire a lot of experience and a lot of understanding for you.

[00:07:14] LA: And a lot of setup, actually, too. It takes a lot of the tracing, working the way you –

[00:07:19] IS: A lot of setup. A lot of setup. Tools, like **[inaudible 00:07:22]** and other players are trying to use, like ATF to make the process a bit simpler. I think that even for organizations that encouragement mesh also make – it's with the pricing much easier. I think, all of those three are only the base requirements that you need to really manage and degree on Kubernetes, to understand what is happening inside cluster. Before I go on to the next store, the most important thing, in my opinion, what do you think? Is it enough, or worth anything?

[00:08:00] LA: Yeah. Certainly, I think, so when it comes to metrics tracing, logging, the easiest to collect, it seems like, and the most easy to put to good use is the metric side, both at APM, as well as infrastructure monitoring. Tracing is really hard to get right. You can get some really valuable information out of it, but it's really hard to get right. I find logging, mostly useful for security, and those sorts of use cases and sometimes post-mortems and things like that, getting timing correct for post-mortems and stuff like that. I really find that when it comes to diagnosing, or pre-guessing when a problem might be occurring, it really can't be both APM and infrastructure metric monitoring.

[00:08:50] IS: I think they favor it. I will say that, what I think that is really lacking in Kubernetes and Komodor is trying and securing solving is to make all of this process of giving an issue, giving a problem in Kubernetes, give you the right amount of data and the right amount of context to solve that issue. One pillar that I think is really lacking in the three pillars of observability is change tracking, or change management. The ability to understand how does your system change over time, across all different pieces that might change. It might be application. In another namespace, it might be a node that's like assuming in the node.

It might be things such as that commit in different **[inaudible 00:09:43]**. I think, because Kubernetes, because modern applications are built from dozens of different tools, you can think about a lot of small Lego like castle and each definitely keep on changing, and everything together create something amazing, which is the system. In order to solve an issue, you need to have the ability to understand what change, where and why and every minute, because, from my friends and if – 80% of all problems occur because someone, somewhere changed something.

I think that having this ability to go back in time, having the system look five minutes ago, when I didn't have teller is super valuable. This was the first thing that Komodor brought to our customers, and we see how to completely transform how the **[inaudible 00:10:42]**, instead of trying and hopping between tools, such as like Jenkins and determining every changes and manual changes inspections, simply have a single place that tells you everything that happened in your cluster in a very easy and intuitive timeline is something that I think it's really a must in today's environment.

I will say that the organization that will completely detox does have some disability already, but it is quite lacking. Even if you have 100% IHT and lead up. That's one thing that I think is crucial when it comes to solving issues and overlap is, do you agree with it?

[00:11:28] LA: Oh, yeah. Absolutely. I talk a lot about correlation monitoring, is what I call it. The idea of what happened at the time when the system started going haywire. By the way, the time when the problem shows up isn't the time that it started. A problem that caused your system to go down now may have started six hours ago, or eight hours ago, caused by something that occurred at that point in time. Being able to go back those eight hours, find the point when the event, or the correlated activity may have occurred and find out what changed during that time is an invaluable tool to being able to solve a lot of different scaling problems in general, but a lot of service-based problems and Kubernetes problems in general. I absolutely agree with that.

It's interesting, because it's not just human changes, too. It's more and more, it's automated system changes. It's auto-scaling events. It's things like that. It might be even service provider events that occur. It's important to capture all of those events, and be able to correlate metrics, traces, logs, etc., with those events to be able to find out which ones occurred at the same time, or what the cause effect may have been between them, etc.

[00:12:49] IS: Yeah. I couldn't agree more. I think that one thing that is really lacking when it comes to troubleshooting, and these are things that it is a game, like a result of the complexity, which gives, that is Kubernetes is the ability to do a correlation, things to understand, not only what happened in your system, but what happens at the same time across different layers, or different view.

[00:13:22] LA: With external system.

[00:13:23] IS: Yeah. External system. Other namespaces, other clusters. Having them on our application, everything might crash your application. You are not the only guy in town older. You are dependent in a lot of different components. It might be other microservices. It might be other clusters, or namespace, or application. It might be resources, such as secret config net and so on. Each one of them might change, and each one of them might be faulty. What Komodor is doing other than giving you a complete understanding of the cluster over time, how do we change and so on, is we try really hard to give you all of the context you need in order to find out the issue, or to solve the issue that you're experiencing.

We know how to do the correlation, and to give you a very clear view of what is currently happening in your system, but also, what other resources and other things that might affect the system happen for you at the same time. We take a lot of the complexity that need Kubernetes and we think we define for our users. That by itself, I would say is very valuable, because Kubernetes is so complex.

[00:14:45] LA: That's interesting. I actually like to pursue that a little bit deeper, if you don't mind. Let's say you have two separate clusters, and they obviously work together, but they're “applications.” I'll use the word application quite loosely there. They're separate entities, but they're both applications in the URL you're working with. They do interact. I think what I heard you say is one of the things Komodor can do is, if application B is acting up, and that it's affecting application A, then you can show the correlation in application A, of any problems that are occurring there of what might be caused by the problems in application B.

I'm assuming that works, because you're running Komodor in both of those systems. What about external systems, like a SaaS service provider, or an infrastructure, AWS service, or something like that? You do the same thing with that? How do you make that happen? Do you do API level monitoring, or what?

[00:15:48] IS: It's a great question. I say that everything is connected, and everything into one another. It is something we know how to support. What we do is we know how to fetch events that happen across other systems. Other system might be things like LaunchDarkly. Maybe you just change the future plan. We also note to show you all of the alerts that happen of the system, because we have a very deep integration with blockchain and PagerDuty. A lot of the time, all you need is to have the issues to investigate and to know that you have an alert, or an external resource just changed an hour ago, or a minute ago. Because we know how to take all of those different things into a single place, it gives our users the ability to get a holistic view of what is really happening in their system.

It is really important to say, take it away, that we are not magicians. We are not throwing a lot of mumbo AI things on it. We do what we do, simply because we are really focusing around Kubernetes, on solving issues in Kubernetes. We know how different resources can be tied to the Kubernetes resources. We utilize our understanding of alerts, issues, problems, and Kubernetes to reduce the noise, and to give you a thorough and complete picture. One of the things that we keep on hearing from our customers, and basically from everyone is the hard thing today, not only to collect all of the data into a single place, but the ability to find relevant metrics, or the relevant changes, or relevant issues, and to reduce the signal to noise. You have only the important relevant things showing up too, which is a very challenging job that we pride ourselves at.

[00:17:46] LA: Got it. Okay, that makes a lot of sense. Basically, you're making sure that you get as much data as you can that you can correlate together. But your focus really is on the Kubernetes side.

[00:17:58] IS: Yeah. Exactly, exactly. I will say, another thing that is becoming more and more predominant in our users, like in the classroom that we try to look for and it is. Like, there are all of the problems that we've just talked about, when it comes to correlation of issues and so on. We now also have a huge, huge, huge experience and knowledge gap. Because Kubernetes by itself, it is still new. Like the project, there are not a lot of technical DevOps, or dev experts in

Kubernetes, on one hand. But on the other hand, we see the shift left movement, we see how more and more companies are trying to move their purpose, or trying to even have the ownership on certain issues and so on. We see organization as a re-platform level that are auto-swamped with issues and with alerts, and they're trying some of their issues into the developer, but the developer don't have the tools, experience and the culture of the time in order to solve the issue.

We have a system that is becoming more and more complex on one hand, and we have users that are quite limited when it comes to solving issues. We see the clash here. We see how hard it is for a developer to understand how everything that we just talk about correlate. Now he can solve the issue. Where should we look? Should we look in Kubernetes? Should we look in PagerDuty? Should we look at the cloud provider? There are so many places to look at them, for example, correlate, just like, we see how these in turn, like the shift has spoken, don’t really address that part, like the troubleshooting part.

Developers now on releasing the code to production, but in a lot of the cases, they are not the owners of solving the issue. It seems like, it's a shame, and you really try to bring a better system in a way that feels the SRE can benefit from it, but also, the **[inaudible 00:20:12]**, or the developers who are unexperienced in Kubernetes. Do you see the same thing?

[00:20:19] LA: Absolutely. I think, we could do several podcast episodes just on that topic alone. A matter of fact, I talk a lot about STOSO, which is a architectural framework for owning services. That's one of the things I talk about in that. Love to have a deeper conversation on that. It's not really part of Kubernetes, or monitoring in general, but it's about ownership and understanding who's responsible and making sure the right people get the information they need to make the right decisions they need.

Let's talk about starting out here a little bit. You mentioned a few times, Kubernetes is new. You're right. As far as a mainstream toolset, it's relatively new. The vast majority of the Kubernetes clusters that are out there today are, I don't know, let's say less than two-years-old, less than one-year-old. They're fairly young applications, or young implementations of the applications. Therefore, most of the people running them are not experienced in Kubernetes. Or there's not a lot of expertise in Kubernetes yet out there. If I'm running a Kubernetes cluster, and

I'm concerned about keeping it running and monitoring it and doing the right things to keep it up and operating, what should I do first, to make sure the cluster stays healthy? What's the very first thing I should do to my cluster, or for my cluster to guarantee it stays healthy?

[00:21:47] IS: Great question. The first thing other than installing Komodor, I would say, that you should make sure that you have the relevant metrics in place, and have all of the relevant alerts. I'm not sure it's a cookie-cutter template that they can say, other than having relevant metrics, and all of those things that you just talked about. Just saying that it doesn't really have the guys are going to implement it. I think that we need to make sure that you have the day of observability. Then if we come down to –

[00:22:23] LA: Is that Prometheus, or is that something else? Is Prometheus enough?

[00:22:29] IS: I think that Prometheus locking, Komodor is there. It's not the question, I think, of the tooling themselves. It's how you use them. I think, it doesn't really matter if you start your journey and you are not that good in troubleshooting, or understanding and solving Kubernetes issues. I don't think it's the worst thing that can happen to you. I think that what you should be focused on is how do you keep on evolving your system? How do you know what areas are blind spots? How do you know how to improve?

Your event release, like your series of production. Things are going to crash. Now the question, and hopefully, are going to fix it. Then the question is, you know how to learn from each failure, to add more and more capabilities, monitoring capabilities, pricing capability, logs capabilities to your system. I think that the failures are going to teach you quite a lot, you're going to understand that maybe you must wait for your nodes, or you must get an alert every time of us just getting changes, or every time core DNS crashes. There are, I would say, endless failure scenarios.

I know it's so true, we spend a lot of time researching Kubernetes expellers scenarios. The important thing when it comes to monitoring your Kubernetes cluster is I think, like you had to say. Take into account at first that you're not going to get the price and your assistant is going to be good only if you will know how to do those kinds of operations, do the post-mortems and keep on learning from mistakes. I think, that's the most important tip, or trick I can give people.

Remember that you are not going to make the price right, your system is going to crash. You need to make sure that the next time you discover it faster and you will be able to solve it faster.

As long as you have this mentality and your system is not super, super, super downtime sensitive. If you're a medical device with a 100% uptime, then don't listen to my advice. That's for any other system. Make sure you have the basics and to be always in a state of mind where you try to improve any price and more and more team, as they think, like the players are going to teach you quite a lot. I really believe in learning from failures when it comes to setting up the system.

[00:25:04] LA: I completely agree with that. I just want to keep shaking my head here. I know that listeners can't see me shaking my head. Of all the failures that I've been involved in and all the different applications that I've owned over the years, over the decades, the most important thing that we've ever done is to do a good solid post-mortem that is not focused on who caused the problem, but what the problem was, and how do we avoid it in the future. Every single post-mortem, I always insist that something is learned from it.

Whether that's something about the application, or something about the monitoring, or the responsiveness, or the system, or whatever it is that you learn something from it, that improves the overall system as a whole, application infrastructure, team, whatever it is, something as a whole gets improved as a result of that problem. If you keep doing that, then failures end up being opportunities, and things work out so much better. I am a big advocate of that. That's great to hear you say that, too.

We haven't talked much about cloud computing at all. Kubernetes and microservices aren't cloud. Cloud is different. Yet, there's a strong connection and correlation and interest. Connecting cloud computing, Kubernetes and microservices all together. In fact, the very popular, currently popular term cloud native ingrains that strong connection between those pieces. Where do you see the future of the cloud with Kubernetes and microservices, where do you see that future headed? Where are we going with use of the cloud with Kubernetes and microservice architectures?

[00:26:47] IS: I really think that we are going to like them. I'm not sure we could see that happen. How it feels like Kubernetes is going to become that cloud. Then today, it has so much community and such good foundation that I see that cloud by themselves will become less important. Kubernetes by itself is going to become your own Kubernetes thing don't really care that much, if it's on top of AWS, or beginning, or something else within it when I think about the future. This is where we're going at. I think that the main blocker at the moment is around databases. Database is huge, stateful application. I don't think that Kubernetes got this covered yet. I do think that as we go, like if Kubernetes will be able to solve it somehow, there's a lot of open source projects trying to help you move stateful application into Kubernetes, I think once these products will be covered, then I really think that nothing would stop Kubernetes from that working in the cloud, and that might be the case.

[00:28:09] LA: That's a great way to think about it. One of the things I spend more energy talking about than I like to is serverless. I'm one of those people that believes that serverless is more hype than value at this point. I think, Kubernetes has the ability – Kubernetes cloud blanket has the ability to replace the need for serverless completely and to the point where you don't worry about deploying applications to servers. You don't worry about deploying lambda apps to serverless. Instead, you deploy containers of various sizes, tiny to large, into a Kubernetes blanket cluster environment. Let that be the way to keep the server abstraction away from the application. I've always thought that was the right way to do it. I've criticized AWS a lot for the way they've done lambda, that it wasn't based on top of containers initially, and it wasn't – they're starting to support it now. It wasn't an extension to EKS, or they didn't extend at least initially the serverless concepts into EKS.

I just never liked the way they did that. It kept those things two separate for too long. Now, they're the different concepts that really don't need to be different. I'm a firm believer with you. I think, the Kubernetes as the abstraction layer for cloud computing is absolutely the right model that we can drive forward in the future. I think, it's going to be the way that we abstract computation in the cloud. It doesn't do anything. Like you say, it doesn't do anything about solving the data problem. It doesn't do anything about solving other communication issues, networking issues, but it does solve the computation problem very, very effectively. I think it's a great start from that standpoint.

Do you see microservices as continuing to be part of that then? Or is microservices – Kubernetes doesn't require microservice architectures, but microservice architectures do thrive on top of Kubernetes. Even if Kubernetes becomes the base and the most popular way of accessing computation, will microservices be the way that we use Kubernetes? Or will that evolve into something bigger and better or different? Or will we just lose track of that?

[00:30:36] IS: It's a good question. I'm not sure. To ground up here, I'm not sure. I think, microservice has lots of benefits, but on other time. We just move the complexity a lot of the time from within the application to the networking and direction for relations to move on. I do love microservices, but I also really hesitate for using them as a solution for everything. I really need a lot of organizational points to form the microservices. I'm not sure. I think, on one hand, lambda makes a lot of sense, right? Having a very safe system, only as a single team, it has a lot of advantages. On the other hand, I think the complexity is real. When you have hundreds of lambdas troubleshooting, developing new ones, is also super, super tricky.

I don't have a concrete idea. I think, and what I see from our organization, we are working with is this, they do have a movement of doing a bit more. I won't call it monolith, but bigger microservices on one hand, but still trying to break things in a normal fashion on the other hand, Not sure. I think microservice on one hand are going to get bigger. Maybe for things that are not core to your application, they will move to things such as lambda, or things like that.

[00:32:13] LA: Yeah. One of the things that's nice about Kubernetes is it's pretty insensitive to the size of the container, right? You can have tiny, tiny containers and very large containers, and they can coexist very effectively, probably more so than you can in lambda. I think that's a fact, that's important, because we talk about microservices, but I've never liked that term. I'm much more a fan of the term service architecture and service-oriented architecture, even though that's a lot of baggage of its own.

I never liked the term microservices, because it implies small. Services don't have to be small. Services can be small, can be quite small, but they also can be decent size and fairly large. There's nothing wrong with that model. I agree with what you're saying there.

We're almost out of time. Before we go on, I'd like to talk a little bit more about Komodor specifically. Now, we've gotten a flavor for what Komodor does from our conversation here. I see you just received a 42-million-dollar Series B round. I think, Tiger Global was the company that led that, was the leader of that round. What are your plans with that money? What's your future plans for where you're headed as a company?

[00:33:29] IS: My plan and our plan in the center with Kubernetes. I think that everyone will run more than one Kubernetes cluster in production, or even out of production, and is going to need Komodor to make his or her life much easier. That means that we are keep on adding more capabilities that needs the same standard – much simpler to use. We allow people who are not experts to solve issues on one hand. On the other hand, we keep on adding more and more data sources and integration. You can have everything on a similar screen. Begin Komodor is super challenging, because on one hand, we want to have all of the data that is related to the show the problems we are trying to solve.

On the other hand, we try to give you only the relevant context and relevant data. Finding the right balance is super tricky. I think, that we are doing a decent job and even more than I have to say that we have hundreds, or even thousands already of happy companies that are using and trying, solving issues in production using our system. Because this number keeps on growing, really is that we hit a nerve. I think that every devops, or SRE that they talk with have so many scars when it comes to having issues and are planning to take it on, and they really make their life easier. The people who are owners of Kubernetes, and who then, a lot of are trying, just time for this and what the hell is just going on here? That's the plan.

[00:35:18] LA: That makes sense. Is this a safe comparison to say that the way you differentiate yourself in the Kubernetes space, is you're more focused on diagnostics than on monitoring? I mean, there are monitoring companies, like Prometheus and New Relic, etc. Your real focus is diagnostics?

[00:35:36] IS: Yeah, exactly. It's not only the diagnostic, to help extend control. We have amazing integration with Grafana, Prometheus, that is of New Relic. I don't try to compete with them. What I'm saying is that you're going to have a lot of Kubernetes issues. With Komodor, you will have less. We also know to find the best practices, and to help you reduce some of the problem. Once you are going to have a problem, I'm going to help you really find the root cause of that problem in an easy manner that both you as the dev can understand and utilize, but also for your liberty.

I think, Grafana and Prometheus are great in being unopinionated to monitoring the system. We take a different approach. Our approach is to work with you every single step of solving the issue. Yeah.

[00:36:29] LA: That makes a lot of sense. Thank you very much, Itiel. I appreciate you spending the time talking with me today. My guest today has been Itiel Shwartz, the Co-Founder of Komodor, the Kubernetes diagnostics platform. Thank you, Itiel, for joining me on Software Engineering Daily.

[00:36:47] IS: Thank you, Lee. It was really fun.

[END]