EPISODE 1502

[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.

[INTERVIEW]

[00:00:45] LA: Cloud native applications utilizing microservice architectures have 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 this pattern enormously. The unique structured requirements of Kubernetes has led to an increased need for Kubernetes specific monitoring and diagnostic tools. There's been a large number of companies who have jumped at this opportunity.

One of those companies is Commodore, a Kubernetes diagnostic platform that focuses on Kubernetes troubleshooting for the entire Kubernetes stack. Itiel Shwartz is the co-founder and CTO of Commodore, and he joins us today.

Hello, and welcome to Software Engineering Daily.

[00:01:45] IS: Hey, happy to be here.

[00:01:46] LA: Thank you for being here. This is great. This is our second time redoing this. So, for those who listened to the first time, hopefully you can hear this one much better, and I want to thank you very much for taking the time, Itiel, to go ahead and doing this rerecording.

[00:02:00] IS: Yeah. I really enjoyed our last conversation. I'm happy that we can do it again. Hopefully, this time, the audience can hear more about Kubernetes, the difficulties, troubleshooting it, and so on.

[00:02:14] LA: Absolutely. Absolutely. Kubernetes is central to operating modern microservice architectured applications, right? So, in your experience, where do most microservice related failures come from?

[00:02:29] IS: Most failure comes from the – there are like two levels of failures, I would say. First one is interdependency communication. Service A is talking to service B, is talking to service C, is talking to service D, like everything need to play really nicely with one another, in order for the application, for the greater thing to work. Every time something around one of those area breaks, then the user is usually left in a situation where he now needs to understand and to find like the root cause in this very complex system.

So, there is the inter application communication that is like one of the very prominent reasons why teams fail, and it's really hard to understand what fail. The other thing is infrastructure itself. Kubernetes, in the end of the day, is here to run your application. You have a container, this container runs some sort of application, and this application satisfies your business needs. So, this is like the basic of Kubernetes. But for this application to work properly, it has a lot of dependencies. It depends on the load balancer. It depends on the persistent volume claim. It depends on the persistent volume itself. Config map, secrets.

So, there are a lot of supporting factors for the application. And again, when one of those things break, when one of those things change unexpectedly, teams fail. Those are like the two main failure scenarios in Kubernetes. I think both of them happens also outside of Kubernetes, and the thing about Kubernetes is that it is very popular, and it is very easy to change things in Kubernetes. That is what makes things even more complex or more complicated, because Kubernetes gives you the ability to change the value of like a YAML file and get a completely different application. In bare metal world, until you start the machine and until you configure everything, until you do the change, it's hours just to configure things. In Kubernetes you change the configuration value, and in 10 seconds, you already have in your application. So, I think that's why it is even more predominant in Kubernetes, but I will say those failure reasons and scenarios are true, for I think, all distributed application, basically.

[00:04:51] LA: Valid point. So, so its dependency management is really the biggest root cause and whether that's infrastructure dependencies, which exist in all apps, or inter service dependencies. And you can argue that Kubernetes applications being microservice based versus bare metal assets tend to be monoliths, have a lot more inter dependencies to deal with, and that's kind of what your point is. There are so many more dependencies involved here.

[00:05:15] IS: I will say that, absolutely, and there is like one another factor in Kubernetes, that makes it more complicated. Kubernetes makes things look really easy to begin. You do the hello world Kubernetes, everything just works. Everything is so simple, like kubeCTL, apply itself to some Git remote file, and you have your application running, and it's amazing. I really love Kubernetes and I think it's like, one, if not the greatest technologies of like the last decade or so.

But when things fail in Kubernetes, you always ask yourself, how did it work until now? The abstraction layer is so good that once something leaks, when you need to understand how did the network policy really work in Kubernetes, until now, you're in a really bad state because Kubernetes lure you in with this very simple to use API, and everything looks like happy. And then you understand that you're way over your head, and I see a lot for the Commodore customers or people that are talking the Kubernetes ecosystem, that the day one Kubernetes is heard, but not terrible. The day two operation, managing the kiosk is really, really hard. Because what we said, but also because a lot of people don't expect it. They think you can throw everything in Kubernetes, and it will work like magically and simply, it's complicated.

[00:06:44] LA: So, it's an exponential learning curve. It starts out very, very easy. But there becomes a magic point where it becomes very difficult.

[00:06:51] IS: Yeah, which is bad. I must admit. It's a really bad like a learning curve, because you don't see it coming. You simply don't see it coming, unless you have someone in the team, who is a Kubernetes expert, and there are not a lot of those kinds of people, and it tells you, "Wait a second. We need to think about how are we going to troubleshoot it. Once you're going to have an issue? How are we going to know what change? How will we know if it's an info problem or an app problem?" You need to ask those questions in advance, so you can prepare your system accordingly. And I will say that not a lot of people are doing so.

[00:07:27] LA: Well, that gets us into Kubernetes monitoring, right? So, what are some of the key requirements for monitoring a Kubernetes stack, and then application in that stack?

[00:07:35] IS: Yeah, great question. For like full disclosure, I worked and I founded Commodore startup design to help people troubleshoot, and to manage chaos inside Kubernetes. And basically, to be on top of what is happening in the classroom. But before diving into like what is Commodore, we say that the basic thing that you must have if you're on multiple Kubernetes clusters, are like the basic. Having something that track your CPU, memory usage, and so on. Basically metrics, it might be Prometheus, it might be Datadog, it might be New Relic, and so on. I would suggest like a centralized logging system, place where you can keep all of your logs so you will be able to search for them very easily.

Again, there is like Kibana and Loki as the strong open source alternative, maybe Splunk. And also, Datadog is like the paid versions, and maybe distributed tracing. Here you have like Jaeger and Datadog and New Relic. This is also very important. I think distributed tracing, as a developer, I really like them. It's fun. You can see the trace map of a single request and understand how did it spread over your system. One request into Cricket, across all of the different microservices.

The issue is from what I see, from customers, and fellow developers, is that distributed tracing is hard to get your hands around, like wrap your hand around. I see a lot of people that are not really utilizing tools like Jaeger and Datadog, simply because it's more complex. So, I would advise it if you know how to use it. If not, maybe you shouldn't do microservices. But if you are in microservices, and you are not using distributed tracing, you're going to have blind spots. So, I'm not sure like where exactly I stand in this area. And the 14th, which I would say, the most important is basically Commodore, and our tool really helps you to understand the current state of what is happening in your cluster, but also the different correlation and dependency that we just talked about. How is my application connected to the infrastructure or connected to other application? But also, we give our developer, our users, the ability to go back in time and to understand how did the application and how did the cluster change over time.

So, you're able to track all of your deployment. You are able to track config map, the change note that was during PVC claims that were unsuccessful, you can go back in time. And basically, you can think about it is like a timeline for everything that happened in your cluster. And when you have a problem, when you have an issue, going to Commodore gives you both all of the history, but also all of the interesting correlation. And we are trying to take the hard parts of troubleshooting communities and to simplifying them. So, I think, for a person, I'm now going and running my own, like Kubernetes shop, and I want to know what I need to install in advance. I will say like, “Make sure you have metrics. Make sure you have logs. Maybe you need distributed tracing, and you most certainly need Commodore.”

[00:10:50] LA: So, it would be in that order. Commodore, of course, but in order you would do metrics first, then logging, then distributed tracing. Is that a fair statement?

[00:10:59] IS: Yeah, no doubt. Metrics are like the basic, and if you don't track them, I think you will be lost. I think for logging, maybe you can work your way around like you do in kubeCTL logs for at least the beginning. If you don't have a lot of application, I think it might work. It's not ideal and there are good open source alternatives, so just install them. But I do see customers that don't have centralized logging, and are able to operate like, in a good manner. I think the thing that is very powerful around metrics, and also like around Commodore, I will say, is that those tools are proactive. You can configure alerts on like CPU 90%, and Commodore comes pre-built in with monitor around Kubernetes, about like the healthiness status of your cluster, your application, and so on. Those teams know how to shove in your face that you have a problem, and it is really important, as that like a DevOps SRE mindset.

[00:12:01] LA: Nothing beats a solid learning process to understand how things work.

[00:12:04] IS: Yep. I tell – again, not a lot of people think about it in advance. I'm going to correct this, how I'm going to know when it fails. But that's the first question I ask myself, start in any dev project, to be honest, when it's going to fail, because I'm sure that everything I write is going to fail. How I'm going to monitor, and I'm going to be able to bounce back really fast. I think that's the mindset that you must have in a cloud native area. Application do tend to fail. The question is how fast you are able to detect it and remediate it, and basically, to fix the issue. So, this is something that first of all, you need the mindset. And secondly, you need the proper tooling, and Commodore really shines here. Again, also, like metric Prometheus is a great tool, like Datadog again. It's also like a great tool. We use that and I’m quite pleased with that.

[00:12:58] LA: So, let's walk through a scenario here. Let's say I'm building an application from scratch. I'm new to running Kubernetes. I'm setting up Kubernetes in production, for the first time. Where should I start to be able to watch and make sure my cluster stays healthy? What should be the first things I do?

[00:13:20] IS: Yeah, I think Commodore, or if you're not using Commodore, then you're probably using kubeCTL, right? That is what you're going to do. You're going to install your first app, and then doing some sort of command of like, giving off the resources and make sure everything is healthy, and is on track. I think that's the basic of working in Kubernetes. And then going to Commodore or to the metric solution of your choice, and having those like – after you validate that everything is working correctly, making the proper alerts in case things are going to go wrong. I think that, this is how I would like start the process. If you're like still not in production, maybe I want to alert at the very first moment and I will like do the iteration until I feel confident enough and only do the alerts. Because if your application is currently crashing, you don't need alert. You know that everything is bad, like you're not going to tell him that. So, you don't want to have this kind of noise is also important.

[00:14:23] LA: Let's start talking about cloud a little bit. Now. there's nothing cloud specific about Kubernetes. Kubernetes runs everywhere, not a problem. But specifically, in cloud computing, the cloud computing ecosystem has really taken a strong interest in Kubernetes, and specifically the microservice development patterns that Kubernetes encourages, right? So, in fact, the term cloud native, which is kind of all the buzz right now, if you will, ingrains the strong connection between microservices and Kubernetes and ties them closely with cloud computing. So, where do you see that connection going in the future? Will microservices continue to be the dominant growth pattern in the future? Or will things change? What do you see happening there?

[00:15:11] IS: I think like a couple of things. First of all, I think that Kubernetes, by itself is becoming the new cloud. I think we can see like really clearly, as more and more customer, players, customers, are going like multi cloud, and they want one abstraction layer to rule them all. I think Kubernetes is doing a wonderful job in like being that –

[00:15:33] LA: Kubernetes is the new cloud API.

[00:15:37] IS: Yeah. The thing that is stopping Kubernetes from owning the market is mainly around stateful application. Things like SQS, S3, RDS, those tools are a solid, and they don't have a good gratis alternative at the moment. Until that won't happen, I don't think that Kubernetes can become the new cloud. I think it's a very big barrier to cross. But there are a lot of different startups trying in aiming to solve that kind of issues. I think, once they are going to be like stateful application on top of Kubernetes, this is going to be a game changer.

So, to your question, I think, it is becoming the new cloud, and I think like AWS is always a bit bipolar around that. Most Kubernetes workloads run on top of AWS on one hand. On the other hand, AWS was the less big company that really had like a Kubernetes solution. Compared to GKE, like Google offering, AWS offering is quite bad, even like, I will dare to say, underperforming a lot of different areas. So, Kubernetes and AWS have a love, hate relationship. I think the main reason is that AWS is a big threat by the popularity of Kubernetes. I think that's quite interesting.

On the other hand, you see Microsoft, Google, IBM, really contributing to Kubernetes. Because I think they see Kubernetes as like a game changer in how people view cloud native application. They want people to think, in Kubernetes clusters, and not in AWS, EC2 machines. So, I think it's an interesting shift that we've seen in the industry. You also asked me about like microservices. I'm not sure. I have some love for like bigger services, not monolith, but not super small. Because once you have a lot of small microservices, then you just move the problem to the dependency between those microservices. That is why I was never like a lambda fan, or like a serverless fan, to be honest.

[00:17:44] LA: I joined the revolution here. I completely agree with you on that. I've always thought that was not quite the right solution.

[00:17:51] IS: There's a lot of bugs here. So, I'm not sure like what will happen. But if you asked me, medium-sized services, is the way to go. Not like functions, but not something super big. It's hard to find the right balance. But they see the industry going there. The industry is trying to go to smaller and smaller microservices. We see projects that are trying to put serverless on top of Kubernetes, like a **[inaudible 00:18:16]**, I think, Open-Fast, maybe serverless as well. There are a lot of different projects in that area. But it's interesting. I'm not sure like what is going to happen. Because if you ask me, there are already too much microservices, and people are not stopping. So, I don't know.

[00:18:32] LA: One of the things I talk about is I've got this thing I call the Goldilocks calculation. It's to make services just the right size, not too big, not too little.

[00:18:40] IS: Oh, interesting.

[00:18:43] LA: But there's nothing wrong with serverless per se, but for the types of things that it's designed for, and the types of applications, and types of services that work well there. But where I have a problem primarily is what I hear companies – I've had companies come to me and say, "Guess what? We built our entire application on top of services. Aren't you glad? Aren't you happy that we were able to do that?" I said, "No, I'm really not. That was a goal. Why would you want that to be a goal? Why wouldn't you want to build your application in the way that makes sense the most practical way that makes sense, for that given domain space, versus force the technology on it." Their goal was to force feed the technology and that just didn't make any sense. So, in my mind is monoliths have their place. But yeah, certainly the old style. Serverless has its place, but it's not everything. And there's this middle ground where I think most services should fit in and that's the Goldilocks zone I was talking about.

So, we've talked about microservices, we've talked about infrastructure and cloud, but we've been kind of assuming that cloud infrastructures were going to dominate infrastructure moving forward, right? We've kind of made that assumption in some of the things we've talked about. Do you see that as the case? Or do you think there's a plateau where cloud adoption reaches a certain amount, and then that's it? Is everything moving to the cloud the way some people predict? Or is there always going to be a place for on premise applications?

[00:20:18] IS: I'm quite sure that there is going to be a place on premise solution. It's a question of money, right? There are two questions here. One is the privacy question. I think, as time progress, more and more companies are able to solve those issues, even in the cloud. We're talking with banks, and so on, like old fashioned Swiss banks, and even some of them are already in a drawer or things like that. So, I think the privacy issue is becoming less and less like dominant on one hand. On the other hand, there is the question of money. When a lot of money is on the table, I think it might make much more sense of running your own bare metal on prem, on some of the workloads. I think it really depends on the type of operation your company is running. We see a lot of companies using Kubernetes as some sort of equalizer between the on prem and the cloud, because in both cases, they are running Kubernetes and it simplifies their operation like burden.

But cloud is becoming much more expensive and people don't really like to pay, right? So, I think that for a lot of companies that have specific use cases, if it's like GPU, or very strong machine, or I don't know, their own use cases, having some sort of the application running on bare metal makes total sense, and I'm surprised to see so many hybrid solutions from within our customers even, because they're using Kubernetes hybrid, they're using Kubernetes on prem and on AWS. They want like one tool to be able to visualize everything and to understand what is happening across those different layers.

There was a time, I was like more skeptical about the future of on prem. But I see that because of the costs here, it's a matter of money. As someone who founded the company, you care about the success of the business, and no one really cares in the top layer of the company. If you're only like bare metal faces like Cloud. They care about velocity. They care about cost. They care about security. They care about a lot of things. But the implementation by itself is not the interesting thing here. That's the case what I'm seeing. But I wonder like, what do you see in the market?

[00:22:41] LA: Yeah, so I see some workloads, specifically, I would say high CPU workloads, and high memory workloads, and long running workloads tend to perform worse on the cloud. So, those – if people were doing a lot of those, tend to get turned off by the cloud. In fact, one of the problems I see is they buy into the cloud hype, look and move into the cloud, looking to build that will come from that, and then regret the decision or back off from the decision and go back on premise. I think, more concerted effort to look at the best utilization makes sense.

The cloud is great for what it's designed for, which is dynamic workloads. And given that most applications, I think, are or can be dynamic workloads. It's a great model for that. But not every workload is a dynamic workload. So, high CPU, high memory, long lasting jobs, long lasting processes, do well on premise, much better than the cloud. But anything requiring any level of variability or dynamicness. One of the things I love about the cloud, we talked about cost a little bit. But one of the things I love about the cloud is that cloud workloads tend to be built into cost of goods sold for a SaaS company, versus cost of infrastructure, which is the color of money is very important when you talk to finance people about where you put things. So, being able to tie your workload to cost of goods sold, vary based on sales, is an incredibly valuable asset and makes your money a lot cheaper and a lot better. I'm right with you there.

[00:24:24] IS: Is the cost part, it's easier on prem or on the cloud, like doing the attribution?

[00:24:31] LA: Well, so it's easier to equate cloud costs with cost of goods sold, because you can vary the usage of the cloud resources based on your sales, right? If you're a SaaS company, there's a strong correlation between the number of people who buy your product and the more people that use your product. Of course, there's lots of variability there. But there's still a correlation there. So, it's very easy to consider the cloud costs to be – since it's tied to revenue, versus tied to just a fixed operating cost, you can vary the expenses based on what you need. And that sort of money that's used for that it's a lot easier to come by. It's very easy to get investors to invest money, to cover the costs of existing kiosk, right? That's a very easy source of money to get. It's a lot harder to get companies willing to invest in tooling, and infrastructure and upfront costs, because that's much more speculative. That's what you need when you're building a data center and an on-premise system, is it's all infrastructure costs. So, you don't build an infrastructure based on your current sales, you build an infrastructure based on potential sales. And so that's variability, means the money that you're putting into infrastructure is much more expensive money to be using than money that you put into cloud computing.

[00:25:54] IS: Interesting. Interesting points about the costs and the analyzing of the costs, so really interesting. I would say, obviously, like Commodore, we're a young startup and are growing fast. Obviously, internally, we use Kubernetes in the cloud, and quite happy and pleased about it. But who knows? Like in a year or two, when we're going to be huge, everything is on the table.

[00:26:15] LA: Make sense. Make sense. So, what's the future of Kubernetes in general?

[00:26:20] IS: I think, I will answer it from like two different aspects. First one is like the technology aspect, and that is, like I said, I think the future lies in stateful application running on top of Kubernetes database as like queues. I think this will be the next big jump for Kubernetes if they will be able to do it. I think it is like technology possible. But it requires a lot of work making Kubernetes work well, for those kinds of workloads, those kinds of application. In a sense, it's like the opposite of the core of Kubernetes, like future your application as cat, not as favorite pet. And a database, most of the time is really like a favorite pet. You have one RDS, I don't know, and you love him, and you don't want him to die unexpectedly.

So, I think it's like an interesting milestone for Kubernetes to overcome, and it will make it much more popular. I think that's one thing. The other thing is around adoption. I see more and more companies moving and migrating into Kubernetes on one hand, but they also see the shift left movement, and I see more and more developer working with Kubernetes. So, companies are migrating to Kubernetes. But inside those organizations, some of the troubleshooting ownership of the services is transferred from the DevOps SRE. You name it to the developers. I feel like this, by itself, is super interesting. I see a lot of customers going into Commodore because we simplify Kubernetes and Kubernetes troubleshooting for the developers, and they try to remove the burden from them. Because it's hard to be an SRE or DevOps. And they try to empower the developers. I think, this is something which we are seeing that keeps on growing and keep on progressing. More and more customers who are developers, and not SRE, and not DevOps. So, I think it's like, overall, like the future of the software industry. And in particular, I think of Kubernetes.

[00:28:27] LA: Makes sense. Makes sense.

[00:28:30] IS: What do you think about this?

[00:28:32] LA: So, I actually completely agree with you. Let's talk about Commodore. Now, you just had a $42 million, I believe, Series B round?

[00:28:39] IS: Yup.

[00:28:41] LA: So, you've obviously see some future in Kubernetes, right? And you see a future in Kubernetes monitoring. Where is Kubernetes monitoring going into the future?

[00:28:52] IS: Yeah, I think first of all, Commodore, becoming the standard factor of like Kubernetes, and understanding Kubernetes clusters. Other than that, I think that two years ago, when we started Commodore, not a lot of people were using APM solutions. The industry as a whole was much weaker in terms of like observability, monitoring, and so on. I think it's becoming the standard to a level that I really wonder, what will happen in terms of cloud providers? Datadog is a huge company, right? Like a really successful company and a product. I don't think AWS are too thrilled about like the success of Datadog or like of New Relic and so on, because they have some sort of monitoring and observability. It's quite bad and this is why people are like dumping it and going to other solutions.

But I think that the cloud providers are seeing that as like something interesting, and they are spending much more resources on that area. So, I think if you'll get everything already baked in inside the cloud management, it will really change how people see monitoring, because it will be so obvious. Obviously, like I just installed the Google Classroom, I have metrics, logs and maybe distributed tracing already baked in. I think it's very interesting to see how it will become more like the standard.

[00:30:14] LA: Yeah, that makes a lot of sense. What about the distinction between diagnostics versus monitoring? In their words, using these tools for alerting when problems occur, versus when there's a problem, diagnosing the problem and solving it? How useful do you see the tools for Kubernetes now in each of those two scenarios?

[00:30:38] IS: Yeah, again, in Commodore, our goal is to help you find the root cause and even to solve the issue from within Commodore, doing actions on our clusters and so on, to complete the loop. I think that the main issue that most metric solution have, is that okay, I have a problem. I understand that I have a problem. But why? And to answer that, why, usually, and you know we see hundreds of customers. You usually need to deep dive into another tool. It might be Kubernetes dashboard. It might be going to GitHub. It might be going to the cloud provider. I don't know. But usually, you can't close the loop from within the metric solutions. I think Datadog are adding a lot of features to try to solve that. They have the logs, they have the metrics, they have the APM. Now, they have profiling, and they have like error tracking. I don't know.

So, they're trying to own all of this. I think that one of the key capabilities of Commodore is currently lacking in this industry, is to focus more around change tracking, and understanding how did your change repel over the system. In the past, there was the change management process, and I would send you a JIRA ticket, and you will do the deployment thing, and everything was super clear, and super strict and super slow, right? But now we're in the opposite direction, everything is super fast, things break. You don't have any idea what is happening. I think that, we will see more focused on that area, like doing what Commodore is currently doing, but I think other players might also do similar things, because it's so valuable. What do you think, I wonder?

[00:32:20] LA: Oh, sure. Yeah. So, I think the use cases for diagnostics and monitoring are radically different use cases. But I think a lot of the tools end up being the same thing. They're just used in different ways and they need different optimization. So, I see a lot of what you're saying. You're seeing

Datadog starting with monitoring, now moving towards diagnostic. And then you see companies like New Relic that started in diagnostic, and then following through into monitoring, and then Commodore, and how it fits into the middle there. I think it's going to be more of a difference in how data is used than different types of data collected. That's going to make the difference between solid diagnostic, monitoring versus solid monitoring, monitoring, analytics monitoring. But that's just my view, but it's going to be interesting to see what happens there.

So, thank you very much, Itiel. I appreciate you spending time with me today. My guest today has been Itiel Shwartz, co-founder of Commodore, the Kubernetes diagnostics platform. Thank you, Itiel, for joining me on Software Engineering Daily.

[00:33:26] IS: Thank you very much. Pleasure to be here.

[END]