EPISODE 854

[INTRODUCTION]

[00:00:00] JM: Sheng Liang was the lead developer on the original Java Virtual Machine. Today, Sheng works as the CEO of Rancher Labs, a company building a platform on top of Kubernetes. Sheng joins the show to discuss his experiences in the technology industry.

The container orchestration wars had many victims. The competing standards for how an enterprise should manage its numerous containers caused several companies to go down a path where they were building infrastructure which eventually had to be replaced. As Sheng discusses in today's episode, the container orchestration wars almost killed his company.

Rancher was originally built on top of a different container orchestrator and the migration to Kubernetes required a massive rewrite of the Rancher platform. The container orchestration wars were not the first technology battle that Sheng has seen in his career and it will not be his last.

In today's show, we discuss the nature of technology wars. Are they necessary? How can a software company minimize the damage caused by a war between competing standards? Sheng was an excellent guest and we didn't cover nearly as many subjects as I wanted to. So, we'll have to do another show in the future at some point.

A few quick announcements; we have a new app for Software Daily on iOS. We will have Android soon. This app contains all 1,000 of our old episodes. If you're looking for a specific topic, like cryptocurrencies, or Kubernetes, or business in philosophy, you can find all of the episodes related to that topic in the Software Daily app. You can comment on these episodes. You can download them, bookmark them. It's a dedicated platform for Software Daily, and you could become a paid subscribe to listen to episodes ad-free by going to softwareengineeringdaily.com/subscribe.

Also, I'll be attending some conferences in the near future. Datadog Dash in New York City, July 16th and 17th; and the Open Core Summit, September 19th and 20th in San Francisco. I'd love to

see you at either of those conferences. I'll actually be emceeing the Open Core Summit. So I hope to see you there.

[SPONSOR MESSAGE]

[00:02:23] JM: DigitalOcean is a simple, developer friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow. DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support, plus they've got 2,000+ tutorials to help you stay up-to-date with the latest open source software and languages and frameworks. You can get started on DigitalOcean for free at do.co/sedaily.

One thing that makes DigitalOcean special is they're really interested in long-term developer productivity, and I remember one particular example of this when I found a tutorial in DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence, and an attention to just getting developers off the ground faster, and they've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That's the D-O.C-O/sedaily. You will get started for free with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:23] JM: Sheng Liang, welcome to Software Engineering Daily.

[00:04:27] SL: Thank you. I'm glad to be here.

[00:04:30] JM: You were the lead developer on the original Java Virtual Machine. Is that correct?

[00:04:36] SL: Yeah, that was a while ago. It was really my first job out of school.

[00:04:41] JM: Tell me about building the JVM.

[00:04:44] SL: Yeah. I mean, it was really exciting. It was my first entry to the computing industry and I could not have expected a better job than that. It was also in the heydays of internet and Java really found its product market fit, empowering the backbone, the application layer of the internet. It was a perfect timing for not just the industry that internet was taking off, but there was also Y2K scare at the time.

So, a lot of the applications were getting rewritten. Not unlike how people retool for the cloud or cloud native for the container today. It's also back then that basically meant you re-architected, re-implement and rewrite your application using Java. So it was a really exciting time, and we did a lot and we learned a lot.

[00:05:40] JM: How was your first job writing the JVM? It's a pretty important piece of infrastructure. Why did they put like a new grad on that?

[00:05:49] SL: Yeah. I mean, I did research in the university building program language compilers. I mean, it was fairly specialized, the virtual machine and garbage collection and stuff like that. When I graduated in '96 and they already had a version of Java running, but it wasn't quite – So I didn't have to start from scratch, but it wasn't quite industry-grade. I remember I was able to crash it very easily just using a simple program I write, and it runs for a few minutes and it crashes. That's basically where we were.

It also ran very slowly back then, because it only had an interpreter and we really had to put in all the just-in-time compilers and code generators and made it efficient. Of course, I was one of the early members, but obviously wasn't really the sole member, and they were other few guys more senior than me and they were also – Then later on, the team really blew up and there were literally dozens of people working on it.

[00:07:01] JM: What were some of the early mistakes that you made when architecting the JVM?

[00:07:07] SL: I mean, I think in retrospect, it was a pretty successful effort and it's difficult. When I started working on that, I think they were at version 1.02. So, it was just after they kind of shipped the first what we deemed as the production-worthy release, and I think I ended up shipping 1.1, 1.2. I mean, you think about – I think JVM is probably at version 8 or 9 now. That was really more than 20 years ago.

The issue back then was I thought we were honestly being too conservative in retrospect. Probably didn't quite even comprehend the amount of adaption it was going to get. I mean, I think we had the ideas that it was – It could potentially be I think maybe the leaders. Some microsystems at that time envisioned it. But we had a very specific ideas about how we're going to implement this underlying technology. So we had to do a lot of tradeoffs, and I think it kind of led to – In the end, we couldn't really get one piece of technology, like JVM technology that met diversing needs. So, then different profiles and different editions started to appear.

Then, for a while, Java sort of accused of leading to fragmentation. There were different implementations of it. As an industry, we all learned a lot overtime. I mean, back then, we actually thought it would have been possible to enforce compatibility with just tests, because that's how people use to do it, with protocols and with storage systems, with networking systems. But it turned out for language and their platform. It was just way too difficult.

These days, I think people are more pragmatic when it comes to things like compatibility. Generally, these days, it's accepted. You kind of almost have to stick to a common implementation to be compatible.

So, another thing, I think Java was also essentially open source. But I think even when I started working on it, that the Java source code was actually publicly available. But I don't think it actually had an open source license. So, like you had to fax something to Sun and then they would basically send you some kind of a packaged source code, like a ZIP file or something. But you don't really get to see what's kind of going on every day, and if you obviously want to use it for commercial purposes, then you have to still go talk to Sun and get a license. So, it's a

very, very restricted form of open source. Overtime, the Java community really started to open up, and these days, practically, all of Java is open source.

So, I think in retrospect, I think Java might have been even more successful had it gone to open source standard implementation, that approach, faster. Because for a while, they were a different implementations, proprietary implementations of Java. In retrospect, I think today, that seemed like not a great business opportunity compared with all the other stuff you can build on top of Java. I think those are probably at least some of my learnings out of that experience.

[00:10:45] JM: You've been through several acquisitions in your business life. You've seen several phases of the software industry. One of those phases was when you were an entrepreneur during the open stack days, and I often hear comparisons between the open stack ecosystem and the Kubernetes ecosystem. How would you compare those two ecosystems?

[00:11:14] SL: Yeah, that's a great question. I do hear about it a lot especially a few years ago. I don't think I hear about it as much anymore, because I think it turned out both of these ecosystems played out in quite different ways. When Kubernetes was just getting started, there was more of a direct analogy and did hear about it.

I always thought it was a very, very – It was not really the right analogy. Even though at a very high-level, at sort of a superficial level, there are some similarities, but fundamentally what the software does and the maturity of the software, the applicability of the software and the way the community was operated. They were just entirely different. So, actually, any kind of similarity or analogy is actually quite superficial.

One thing that always impressed me with Kubernetes is as much as I loved working with open stack and those type of technologies before, it just didn't come close to the level of adaption and maturity as Kubernetes. So, I saw a lot of people in the early days struggled really with the installation and operations of open stack, and these things were a lot less of a problem with Kubernetes. Just the industry and the community has matured to such a degree that the users and the vendors were – We felt a lot more welcome as we adapted Kubernetes technology. So, it was actually a very, very different experience.

[00:13:12] JM: Rancher Labs is the company that you started. You started in 2014. What was your original vision for the company?

[00:13:21] SL: That's a great question too. As you pointed out, I mean, we had quite a bit of an experience. The founding team of Rancher Lab, myself, and Shannon Williams, and Will Chan, and Darren Shepherd. We all had experience working together initially at cloud.com that developed cloud stack, and we worked with open stack, and we got acquired into Citrix and sold a software, open source software platform that helped a lot of people build public clouds and private clouds that sort of looked and operated like Amazon EC2. So we did that for many years.

What we learned out of that was that approach had limitations, because these infrastructure clouds that were getting built weren't really a commodity. They weren't really interchangeable, like every open stack cloud. Even if they all based on open stack technology or cloud stack technology, that was what we were mainly working on. They were still different, because there's no de facto standardization. I mean, to some degree, the de facto standardization in cloud is AWS, but now you can only get that from Amazon.

Even though Amazon, Google, Azure, they can all run intel-based virtual machine images, but the image format is different. The API is different. All the services around it is different. So, cloud infrastructure became contrary to our desire. I think most of customers' desire. Cloud infrastructure became probably the least commoditized and most depreciated type of infrastructure. That was very unexpected for us, because as you probably know, the term Rancher came from this saying pets versus cattle. We always believed in a cloud world, in the cloud native world. Infrastructure should be treated like a resource, which means they should be largely interchangeable. They should be largely a commodity.

In fact, in some sense, the cloud infrastructure became less of a commodity than before. In the old days, I could take, I could buy a Dell server, or I could buy an HP server, or I could buy an IBM server, and these things might be priced a little differently. They might have their own distinct features, but none of these really impacted the customer's ability to interchange them and essentially treat them as commodity, and these vendors really had to – Essentially, to a large degree, differentiate on price and services.

Since that's not really the case with cloud infrastructure, which kind of almost goes against – So it's almost like you pay what you use or you treat your computing resource as air or water. That's like a resource. Except there's only one place you can get it. So, it was never really ideal from our perspective. That's what really got us into containers and Kubernetes, because, finally, with Docker and Kubernetes, there's a software packaging technology as well as infrastructure deployment and infrastructure abstraction technology. There's infrastructure platform technology that you can count on that works everywhere.

Today, you can get Kubernetes clusters from Google. You can get it from Amazon. You can get it from Azure. You can run it on VSphere. You can run it on bare metal, and there are just so many ways to run it. We obviously built a great business just helping people doing that. But the benefit is now, finally, the infrastructure is entirely consistent and it's consistent in ways that's very fundamental. It was not like the java type of fragmentation back then, which we mistakenly relied on some kind of a compatibility test suite.

Whereas Kubernetes, these days, you always hear people say, " I use upstream Kubernetes." You not only have to use Kubernetes, but you have to be really honest in the sense you can't fork, you can't – Maybe you can apply a bug fix here and there as long as it absolutely doesn't change any behavior. But, really, it has to be upstream Kubernetes, right? It just made it so consistent. So this is really fantastic. I mean, we feel like, finally, the vision of cloud infrastructure, the vision of pets versus cattle, that finally could be fully realized. That's why we started this company called Rancher. So, enterprise IT can really start to treat all their infrastructure as commodity resources, and they can focus on developing applications and deploying these applications in whatever environments they want.

Anyway, that is really the story and the motivation of starting this company, and we're so lucky that the Kubernetes technology and ecosystem grew almost exactly, perhaps even better than we had hoped, and we really benefited.

[SPONSOR MESSAGE]

[00:19:15] JM: LogDNA allows you to collect logs from your entire Kubernetes cluster in a minute with two kubectl commands. Whether you're running 100 or 100,000 containers, you can

effortlessly aggregate, and parse, and search, and monitor your logs across all nodes and pods in a centralized log management tool.

Each log is tagged with the pod name, and a container name, and a container ID, and a namespace, and a node. LogDNA is logging that helps with your Kubernetes clusters. There are dozens of other integrations with major language libraries, and AWS, and Heroku, and Fluentd and more.

Logging on Kubernetes can be difficult, but LogDNA simplifies the logging process of Kubernetes clusters. Give it a try today with a 14-day trial. There's no commitment. There's no credit card required. You can go to softwareengineeringdaily.com/logdna to give it a shot and get a free t-shirt. That's softwareengineeringdaily.com/logdna.

Thank you to LogDNA for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:20:38] JM: In 2014 when you started the company, that was before Kubernetes became popular. Everybody knew that we wanted a better abstraction to manage these resources, whether it's Docker containers, or VMs, or whatever the abstraction of choice for our servers would be. We all knew that we need it to give it to enterprises and we weren't quite sure how that would look. The way that this played out in the industry was, from my point of view, the container orchestration wars, there you had different container orchestration systems saying, "This is the way that we should be managing all of our containers, or all of our – Yeah, all of our containers basically, or in the case of Mesos, your containers and your VMs," and there were conflicting views on what the API surface should look like, what the community should look like, what the vision should look like, and that's why we had these different container orchestrators.

My perspective of Rancher is you were not totally agnostic of the container orchestrator, but you had a vision for the UI layer, and the user experience, and the enterprise experience. So, the container orchestrations wars for you were problematic, but not the end of the world, and you probably knew throughout that process that you would either need to eventually support one specific orchestrator or you need to support multiple ones, but you would make it through the

orchestration wars okay. You were not in the position that some of the companies that have really bet big on their own container orchestrator were in. Can you tell me your experience of the container orchestration wars?

[00:22:30] SL: Yeah. I mean, it's so amazing. It hasn't really been that long, but it almost feel like [inaudible 00:22:35] now, because Kubernetes had become so dominant. Actually, when we got started, Kubernetes was already launched. I mean, it wasn't by no means ready, but it was clearly going to be – We knew it was clearly going to be something that's quite special. As primitive form as it was at the time.

[00:22:57] JM: Why was that? Why did it look special even in its primitive form?

[00:23:00] SL: Yeah, you could tell from just the general competence of the work and of the people behind it. I'm not just saying – Obviously, the fact is Google – I mean, I think that helps, but it's actually a little more than that. There is a genuine level of competence that just comes through. I mean, there are problems too, right?

I think it was pretty clear that it was going to be something that's very significant, even though it wasn't quite there yet. Either way, I mean, it was very kind of you to say that we didn't get killed by the orchestrator war. But I'll tell you, it was a huge struggle for us in many ways and it almost killed the company, maybe not in ways you think.

So there were a number of problems. One problem was because people didn't know what technology was going to wing out, then a lot of customers were just not moving forward. It kind of paused the market. So that was very bad. There was also another problem, was really, none of these technologies actually worked well. The technology that really worked the best at the time was Apache Mesos, but that was actually Mesos in marathon, but that was actually kind of also happened to be the least complete. It worked to the degree it worked.

In order to make it work, we actually kind of have to do a lot of work on top. Then we didn't think it – I mean, we actually thought that Apache Mesos for a variety of reasons is probably not going to be the winner for this war. So, it's like if we spent a whole lot of effort on it, it just didn't seem like the right thing to do. So that was a huge challenge.

You know, what we did was like a dozen or so other players in this space. We actually ended up developing our own orchestration layer for orchestration platform for Docker. Very few people – There's still a lot of people using it, and for the last couple of years, we've been trying as hard as we can to migrate those folks to Kubernetes, and that orchestrator was called Cattle, and a lot of folks in the industry still remembers that fondly. I mean, it's just something we had to do, because otherwise – Really, people were just trying all kinds of stuff. They were trying to orchestrate containers using Chef, using Puppet, like anything, any tool that could be used. We just really needed something. So that was very hard.

I think, also, the other approach we took was we said we're going to support – Like you said, we're going to support anything and we're going to try to add value on top, whether it's user interface. We had an application catalogue, which was sort of – I mean, these days, people talk about Helm and it's not even a problem anymore. But back then it was like a big deal. You can one click deploy applications, and we figured out authentication. Just really – Environment management. You have these different clusters and who can access what cluster.

So, we owned a number of avenues that we could add some value and build an early business and got a product out of the door that got some very early product market fit. Fundamentally, we knew that that vision wasn't – That was just not a stable state. The technology space, especially open source technology, tend to be the winner take all.

So, what came out of that is when we developed our 2.0 product truly centered around Kubernetes. We basically had to throw away almost all of the 1.X code. So, it was literally a reset. That's why these things almost killed companies. I mean, you could imagine, if that's the impact on the product, that obviously has a ripple effect on the customer-base, user-base. So, it's really very significant. I think a lot of people don't appreciate how tough it has been for this industry because of confusion around the orchestration.

[00:27:29] JM: We know some companies that it has really, really damaged the orchestration wars, and it's just unfortunate. I mean, the benefit of the way the technology industry works in the regard of the container orchestration wars is that you have a marketplace of ideas, and the ideas play themselves out in the open. But the downside is that there are these casualties.

There are these just waste that goes into the lack of central planning. The fact that we have this anarchy that leads to the creation of the best solution, the best thing winning out. Is there any way to avoid that anarchy in the community? Is a community centralized around solutions more quickly?

You see this with Istio stuff. I've been covering the service – I don't know if you would call it a service mesh war. I've called it that kind of as a matter of linkbait, because I'm a journalist. I'm just trying to linkbait people to my service mesh war coverage, but it does feel a bit like a war. There's a war between Linkerd and Istio and probably one of them is going to win, maybe multiple winners. But it feels like the container orchestration wars. That's maybe a tragedy, maybe a feature. Tell me your perspective on the value of wars.

[00:29:01] SL: I think it's great. It's actually – It will happen. I mean, it's unavoidable, and completion makes everyone better. What's interesting is a lot of times the wars may not result in anything in retrospect of substantial value. There's actually sometimes people lose the big picture. For example, I've actually been personally involved in a war myself, actually multiple ones.

I mean, in the Java days, there was Java versus Windows and that kind of stuff. Then when I worked on cloud stack and open stack, there was cloud stack versus open stack. What really happens in all of these situations was the benefit of – I actually kind of – I think both Istio and Linkerd should really thank you for the journalist and the public for describing it as a war, because it actually raises, dramatically raises the profile and awareness of these technologies. In the end, what's going to happen is if the product market fit isn't really quite there, then it really doesn't matter. Nobody wins.

But if the product market fit is there, then it's really interesting, and something like Kubernetes will happen. Then actually the strongest technology and the best rung community will actually win. That's really great. It's almost out of necessity to go through that, because what's the alternative? I mean, the alternative would be something that completely develops under the radar, and then all of a sudden it became a runway hit so quickly that competition didn't even realized that.

I mean, it happened. It happened a few times. It happened with VMWare. It happened with AWS. So that's kind of the alternative, and it happened – You could argue with some great technologies like, say, Elasticsearch. It was just developing, developing. Then it just became a leader. So that's kind of the other alternative. I mean, I would think the second alternative generally results in a much more lucrative business, because now you basically have something that rule the business. Whereas if you look at how Kubernetes played out, is because in order to win, Kubernetes not only had to be the best technology, but it also has to be the best open. It has to be the most open.

With all that competition, a lot of the profit is also competed away. If you now look at the Kubernetes services, like GKE, and EKS, and AKS, I think all of them pretty much sell that service either for free or at a very, very low cost. That's a very big different. There's a very big different, say, from a Kubernetes, cloud Kubernetes service or, say, a cloud database service, like RDS

So, it's really interesting because it's fundamentally such an open piece of technology, with open community, that's openly competed. So that's why like with us, like we have a very pragmatic view about this. We probably recognized earlier that most people that Rancher has been a certified disjo Kubernetes. But unless the disjo is truly, truly differentiated, like k3s, because it's the smallest, it's the easiest to manage. Otherwise, if you just take the upstream Kubernetes code and then say, "I'm going to be a Kubernetes company because I've got 10 certified Kubernetes engineers." I mean, it's not going to be easy for you to make money. So it's not going to be easy to be able to viable a business if you're just using this industry standard technology that you don't even know, or nobody owns at this point.

It really has some interesting implications. But I think it's great. So you just have to be aware which situation you're in. Are you more in the AWS, VSphere, Elasticsearch kind of situation, or you're more in a Kubernetes kind of situation?

[00:33:31] JM: What kind of situation does service mesh seem like to you?

[00:33:35] SL: I think it's closer to a Kubernetes kind of situation clearly. At this point, it seems to me that we have some firsthand knowledge about this ourselves. We don't produce – I mean,

we're not one of the producers of service mesh technology. There's Linkerd. That's part of CNCF. Istio, it's led by Google. But we consume these technologies, because we have to kind of support these things for our customers and we see that Linkerd by being part of CNCF. It's just a more open community. But on the other hand, Istio is just more feature rich. Then you got the power of – The credibility of Google behind it. So, I think we're in a very interesting situation with service mesh.

[00:34:31] JM: So, I guess I'm not sure exactly what you mean by the it's more like a Kubernetes example. Are you saying that we're waiting to see like the – There's going to be some third dark horse competitor that's going to satisfy the middle ground between these two technologies?

[00:34:46] SL: No. No. No. I think in the long run, one of them has to –

[00:34:50] JM: One of those two.

[00:34:51] SL: Yeah.

[00:34:52] JM: It can't be both, right? It's no way.

[00:34:54] SL: Yeah. So I say those two situations, one situation is a piece of technology or service or something that's able to develop pretty much under the radar that all of us [inaudible 00:35:05] such a lead in the industry that nobody is able to catch up anymore. So, I use Elasticsearch as an example, and AWS as an example, or you have another situation where pretty much early on, people realize there's an opportunity and you have multiple technologies, and they all have to compete. But in that scenario, these technologies, because it's competition, they'll actually – They'll maturate much faster, because there's competition. But in the end, still, only one should win.

[00:35:39] JM: I mean, I wouldn't blame you for not wanting to place a bet on either one. It's very hard to tell at this point. I mean, I find this battle to be quite riveting. I've never been much of a sports fan. It's such a cool competition, because you really do have this – You summarized it quite well. Istio, this very feature rich open sourced, but kind of closed and like where did this

thing come from? What are all these partnerships that you're talking about? Versus kind of the dark horse Linkerd that identified the space quite early. I think your analogy there is like Linkerd is kind of like Elasticsearch, where they spotted this space early. They placed a big bet on it and they've just doubled down relentlessly.

[00:36:29] SL: Yeah, let me clarify that. I think, probably, Elasticsearch was in a very interesting situation. It developed over a long period of time where there's no competition at all. So, pretty much was able to take over market share before a viable open source competition even appeared. So, I think at this point, either Linkerd or Istio to me unlike Elasticsearch anymore, because they're just receiving so much attention at such early stage. So, they have to compete.

But the benefit is I think both of these technologies are evolving very quickly. They not only have to evolve quickly in terms of technology. They also have to be reasonably open to stay viable. So it's fascinating. For us, we're more like where Rancher is these are key-enabling technologies for our customers. But we don't really need to place a strong bet on either of them. But at this point, it's very clear though. We're definitely seeing more customer demand for Istio I think just because – Mostly because it just really works, just more powerful. So, as a result of that, we've just put a lot more focus on Istio, but we're not – There's nothing against Linkerd.

[00:37:57] JM: Right. Right. Well, I hope this war doesn't kill your company. I don't think it will. I think you're okay this time.

[00:38:04] SL: Yeah. People have learned. People have kind of learned their lessons. Microsoft just at KubeCon announced service mesh interface. So, I think that's a great initiative that –

[00:38:15] JM: I think so too.

[00:38:17] SL: We support it. We supported their launch, and we have a project called Rio. It's a micro-PaaS. So it's kind of designed to package a lot of technologies like service mesh and KNative. It can run on any Kubernetes cluster. Create a really easy to use PaaS experience for developers.

[00:38:43] JM: That's great.

[00:38:45] SL: In that kind of scenario, because the developer is not really directly interacting with Istio or Linkerd, it doesn't really matter to them what we use. As long as it works, right? We would honestly – I mean, today, Rio work integrates with Istio, because it does everything we need and it does everything we want and it's rock solid. It works really well. But we wouldn't mind one day, let's just say, just say if Linkerd – Let's just say maybe it's smaller or lighter weight or something. Then that could be an option too. The great thing about service mesh interface is it almost make it easy for this thing to become a plugin.

[SPONSOR MESSAGE]

[00:39:40] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take-home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:42:00] JM: The container orchestration wars and the service mesh wars, these are wars that occur at one layer of the stack. But I think at a higher level, there is a different kind of war, and that war is one that you're fighting to win enterprise deals. So, I like to walk around the Expo Hall at KubeCon, and I walk around the Expo Hall and I see, "Okay, there's Rancher. There's Platform9. There's OpenShift. There's Google. There're all these different companies that have built some kind of platform on top of Kubernetes. That is a much less winner take all space than which container orchestrator is going to win. Because every enterprise is going to need one of these things, and if you just think about the fact that, "Okay, right now we are in the early days of enterprises becoming distributed systems companies." There's going to be a whole lot of opportunity both now and in the future for companies such as yourself.

I'm wondering how that affects your long-term strategic vision, because I think of it kind of like Android phone manufacturers, where there's – I don't know how many companies that have built successful mobile phone businesses off of Android. Probably it's in the hundreds. Maybe that's too many, but it's a lot. There're a lot of Android devices out there.

The Kubernetes ecosystem kind of feels like that to me, where it's like you don't have to produce all the smartphones in the world to win as an Android manufacturer and you don't have to capture all of the enterprises in the world to be a successful Kubernetes platform vendor. But you'd like to maximize that capture. So, how do you navigate this highly competitive world of Kubernetes platform providers?

[00:44:03] SL: Yeah, that's perfect. I mean, this is so – The way you characterized it is just so accurate and fantastic and clear. I'll tell you, it's a tough space, because Kubernetes is just like

building mobile phones. It's a tough space, but it hasn't stopped a lot of people being very successful. In the end, it's the customer who benefits.

But I'll tell you right now, the IT guys are really facing some very – As a mobile phone customer, I'm basically pretty happy, I'll tell you. Maybe the reception is still not where it needs to be, but like everywhere in the world, but that kind of stuff is really hard to solve. It's orthogonal to Android and iPhone.

Generally, these platforms are pretty good. Whereas, if you look at enterprise IT, earlier I was saying where should the enterprise IT go? They generally want to move to the cloud, but which cloud? What's going to be their new platform if all these clouds are different? So that's why we fundamentally believe that the IT computing platform of the future will be based on Kubernetes or something like Kubernetes. I mean, for now, Kubernetes is really the obvious choice. So we believe in 5 years, 10 years, Kubernetes will basically be powering all the significant enterprise workload. I think that the industry is moving very quickly toward that.

Now, if you kind of take that for granted, then there's huge amount of challenges these organizations face. I kind of just tell you three. By the way, different vendors, they, again, approach it differently. But we choose to address few challenges that we see. The first challenge is it is just very hard to actually operate Kubernetes. It still requires way too high-level of technical expertise. So, that's kind of almost like the level one. You see all kinds. I mean, Rancher tries to solve it. Platform9 tries to solve it. OpenShift tries to solve it. There are dozens. I think maybe over a hundred Kubernetes disjo vendors try to solve it.

So, this is like the first challenge. I think this is a hardly contested space, and you have to be like truly differentiated, because otherwise if – The way it's going is if people go into the cloud or they go to Google, there's no reason not to use GKE. Why bother paying a vendor for a separate tool that they have to then operate versus just get Kubernetes as a service? It really has to be special, and that's where Rancher is something that's truly special. We developed, really, an industry first, this k3S project. Have you heard about that?

[00:47:03] JM: I have not.

[00:47:04] SL: Okay. It is an extremely small, extremely lightweight Kubernetes disjo that's designed to work on the edge, everywhere, in branch offices. Basically, anywhere that the Linux will go, Kubernetes can go. It can run on Raspberry Pi. We're basically saying if you're in the cloud, then use cloud-hosted Kubernetes. We don't even think you should be running your own Kubernetes clusters. But if you have any reason still be managing your own hardware, which is really mainly around the edge.

We have people nowadays looking to run Kubernetes. In Chick-fil-A, they wrote an article about using Rancher to do that last year in their stores. We were working with the second largest wind turbine manufacturer in the world. They're looking to using Kubernetes to actually run on their power generation plants, which are in the middle of a desert or the middle of a mountain, or oil platforms and energy use cases. So there's still – Like I'm saying, this problem is not solved, and there are tons and tons of user cases that we're just seeing today that people literally want to take Kubernetes everywhere.

Kubernetes, we really believe is going to be just as ubiquitous as Linux and you may not even know that Linux is what's powering your phone or powering your TV, but it's there, right? Kubernetes will be something very similar to that, and there's just tons of opportunity. So, we developed k3s. It really took the industry by storm. It's the fastest growing wind source project in the history of Rancher. I think we hit over 7,000 GitHub stars just in a master of weeks after the project is launched. So, it's just showing no signs of slowing down. So this is truly exciting.

Then, the second challenge we see is, as you know, Kubernetes originally came from Google Borg. So they'd wrote a paper about it. Even better in this case, they actually produced open source implementation of some of the Borg concepts. The whole idea, the original idea was like it's a very big resource pool.

Google as a whole has won Borg. All their resources around the world is pooled together. So, really, if you kind of apply that logic, you should just have one Kubernetes cluster. Every organization maybe should just have one Kubernetes cluster.

In reality, we're not really seeing that, because nowadays it's so easy to get Kubernetes clusters in the cloud. It's so easy – We were talking about running Kubernetes clusters on the edge, on

Raspberry Pi. It's also becoming easier and easier to setup your own Kubernetes cluster. So we're basically seeing these clusters popping up in an organization everywhere, and they need a platform that can provide IT needs control, needs visibility. Needs to apply security policy. Needs to maintain consistency. So that's what Rancher server does.

So, we have a product called Rancher Server, which is also open source. All our products are open source, and that allows you to manage – Really, the true value is multi-cluster management, which also implies hybrid cloud, multi-cloud management. But the idea is it is the Rancher or the catalogue that is Kubernetes. So I hope that makes sense. You have a Rancher server there that manages all your resources, deploys all your apps across wherever these Kubernetes cluster are. So that is actually our best selling product now. That is like 90 plus percent of our business, is just selling Rancher. So that's a big part of our business.

Then the third one we're also seeing is saying Kubernetes is getting adapted everywhere. But Kubernetes was historically designed really for dev ops, even in that case, for fairly advanced dev ops teams. It came out of Google. It has some advanced concepts. The YAML files are not necessarily the easiest thing to compose. There's a bit of a challenge for a lot of developers who are just more focused on building applications to actually directly consume Kubernetes. So that's why we're building this micro-PaaS called Rio, as in Rio de Janeiro, Rio. That allows us to package together technologies like KNative, like Istio, like Linkerd and it can – It's micro-PaaS. It's not necessarily big PaaS. As a developer, I can have k3s running on my laptop, and then I just run Rio on top of that, then I can deploy my applications very easily without even understanding anything about Kubernetes or Istio, but I know exactly how my app is going to deployed and upgraded and tested.

Then I figured out how everything works, I fix my code and make sure it still works. Then I check my stuff in. Then it goes through CICD, which then the whole thing was deployed in exactly the same environment in CIDC. Then, finally, they'll get, at some point, promoted into production as part of that process. So, that is really cool.

So you can kind of see Rancher, where we're really just focusing on how to leverage this Kubernetes technology. How to take advantage of this Kubernetes everywhere vision that we

100% believe is going to happen and how to really build these tools that make our customers successful.

[00:53:01] JM: Indeed. I want to talk a bit about your background and the dynamics between U.S. and China engineering as we wrap up. You got your undergraduate degree in China. You got your Ph.D. in computer science from Yale. I've become fascinated with Chinese engineering, because I went to KubeCon China. I read Kai-Fu Lee's – Did you read Kai-Fu Lee's book?

[00:53:29] SL: I've heard about it. I think I know what it is about. I haven't read it.

[00:53:32] JM: Yeah. Well, that made me curious, because I don't know how completely accurate it was. But he talks about this 996 work style, which is like really appealing to me just because I like technologists who work really, really hard. I admire them.

Another data point that I found interesting was the – You probably saw the Zoom IPO, and I think the Zoom video conferencing software. I think it was built mostly by Chinese engineers and it was led by a Chinese immigrant entrepreneur. So, just a really interesting success story. How do the engineering cultures, the software engineering cultures of the U.S. and China differ from one another?

[00:54:17] SL: Unfortunately, there's not – We don't have a huge engineering presence in China. There are a small number of engineers in China who we originally hired to mainly serve the customers in China, and some of them have been making contributions to the product as well. So I have a little bit of visibility. Probably not as much as a company like Zoom with half. But I just think I have a huge amount of respect for hardworking Chinese people generally are.

I mean, I think the way we grew up, all of us basically felt that being able to work and earn money – I mean, you don't even really have to have any prospect of being successful, but just being able to work and then either directly earning money with your time or just your earning experience. You're making yourself better. That is sort of almost motivation alone. That benefit is sufficient to just work.

I think I'm no sure even 996 accurately captures the work style of Chinese. At least when I was remembered and what I've observed and how I work, you almost spend every waking hour working. So, maybe even sometimes in your dreams. It's literally like, and that is I think what contributed to the – More than anything else, contributed to the rapid development of the economy in China, across the board.

But that said, I certainly would have had the – I mean, if I really wanted to do it, I could have built an engineering team in China and then based all of our development in China. I didn't do it not because I did not like China. Because at the end of the day, all things concerned, is still far more competitive, far more efficient in doing what we do. With all that 996 and everything, doing everything for me – We're doing literally like 98% of the working for the U.S. So that is till – I'm doing it because we're a startup, we're very capital constrained as you can imagine. I go to startups.

[00:56:54] JM: And it's fun.

[00:56:55] SL: Yeah. We always have to do it in the most economic way. But there's still nothing that actually beats – There's still something that – Like at least in doing what we do, that it's very hard to do in China, because it is highly creative work. It requires a lot of passion. I don't expect people to stay in office 12 hours a day. That is honestly not good. I mean, I don't expect people to even do that in China.

But what I've seen is some of the best work I think happens even for me or for you or for a lot of our engineers. When they are at home, not even at work, and then some great idea will come to them.

[00:57:39] JM: Right. That's what's misunderstood about the 996 thing.

[00:57:43] SL: So, I'm not sure – I think you got to take this Chinese work ethic as more of just work ethic than anything else. I actually think the U.S. system. At Rancher, we pride, part of our culture is being extremely accommodating to employee's needs, really. I think we have the highest – I think it's amazing, At Silicon Valley office, we actually have more female engineers

than male engineers. I think one of the reason we can accomplish that is because we're so accommodating to people's needs. A lot of them are working moms. They have multiple children. Literally, they are far beating the productivity of the 20 engineers that work 24/7 in China, but it's amazing, and it's not really just because they're more capable. There's just something about the creativity, the motivation, the environment. It's a very interesting situation.

[00:58:46] JM: Sheng, there's so much we could have covered that we didn't get into. Are you going to be at KubeCon San Diego?

[00:58:51] SL: Yes.

[00:58:52] JM: All right. Well, maybe we can do an actual in-person show there.

[00:58:56] SL: Absolutely.

[00:58:57] JM: Okay, Sheng. Great talking to you. Have a wonderful weekend, and thanks for coming on the show. Been great talking to you.

[00:59:03] SL: It was awesome talking to you as well. Have a great day. Have a great weekend.

[END OF INTERVIEW]

[00:59:10] JM: FindCollabs is a place to find collaborators and build projects. The internet is a great place for communicating with each other. Why aren't we building projects together? Why is it so hard to find other people that will work with you on a new project reliably? Why is it so hard to find cofounders to start a small business with?

We started FindCollabs to solve these problems. I love starting new projects, software platforms, musical songs, visual art, podcasts, and when I create these projects, I want to share them with people. I want people to be able to follow my progress, because as they see that I'm dedicated to consistently delivering quality work, maybe they'll be inspired to join me, and it doesn't have to be on a financial basis. It could be just to collaborate and to make artistic progress.

You can see my profile on findcollabs.com by searching for Jeff Mayerson. You can see the different projects I've made; games, podcasts, open source software. Some of these projects are incomplete. Some of them are well-developed, and this is how innovation and invention happens. Projects start out in an incomplete state, and if you can prove to the world that you work on projects with dedication and consistency, you will get other people to join you. Whether you want to hire them or if you just want to collaborate casually.

You can see my profile and many other people's profiles, projects that they're posting on FindCollabs, and I would love to see your projects there too. You can check it out by going to findcollabs.com, F-I-N-D C-O-L-L-A-B-S.com. You can post a project that you're working on no matter how incomplete it is.

The best projects in the world start with an inspiring vision, whether or not they have code attached. If you like to compete, the FindCollabs Open is for you. The FindCollabs Open is a $2,500 hackathon with prizes for the best React JS project, the best machine learning project, the best game. There are lots of prizes, and you can check it out by going to findcollabs.com/open and post your project.

Thanks for being a listener to Software Engineering Daily and I hope you post your cool ideas on findcollabs.com.

[END]