EPISODE 989

[INTRODUCTION]

[00:00:00] JM: The ride-sharingud
infrastructure at Lyft has a high-volume of traffic that is mostly handled by servers on AWS. When Vicki Cheung joined Lyft in 2018, the company was managing containers with an internally built container scheduler. One of her primary goals in the company was to move Lyft to Kubernetes.

In today's episode, Vicki gives an overview of Lyft infrastructure and the core engineering challenges within the company. One subject that she touched on was the network communications between the user on a mobile device and a cloud backend, and this was a topic we explored in detail on a previous episode about Envoy mobile with Matt Klein. The challenge of having network devices, mobile phones in this case, that have network connections that may be unreliable creates interesting design challenges all throughout the system.

Vicki also discussed the broader Kubernetes ecosystem, because she often goes to KubCon, I've seen her give talks there. She also discussed her time at OpenAI, where she managed infrastructure deployments for scheduling large machine learning jobs.

We have partnered with SafeGraph for the SafeGraph Data Hackathon Challenge. We're giving away $4,000 in cash prizes as well as SEDaily and SafeGraph swag. SafeGraph is a geospatial data company which curates a dataset of more than 6 million points of interest, and SafeGraph provides a high-volume of location data. You can use that data to build apps and data science projects with that data, and if you've been looking for a creative opportunity to explore large datasets, particularly with the potential to win $4,000 in cash prizes, this is a great opportunity. The hackathon is hosted in FindCollabs. You can enter by going to findcollabs.com and signing up.

[SPONSOR MESSAGE]

[00:01:59] JM: Our thanks to Datadog for sponsoring this episode of Software Engineering Daily. In their latest report on Container Trends, Datadog analyzed 1.5 billion containers and found that in orchestrated environments, container lifespans averaged about one day. To easily monitor this dynamic infrastructure, use Datadog's container specific monitoring features. The container map provides a bird's eye view of your container fleet and the live container view searches groups and filters your containers with any criteria, like tags, and pods, and workspaces, and you can try Datadog in your own environment with a free 14 day trial. You will receive a complimentary t-shirt. Just go to softwareengineeringdaily.com/datadog to install that agent and get a complementary warm, comfortable, cool t-shirt.

[INTERVIEW]

[00:03:02] JM: Vicki Cheung, welcome to Software Engineering Daily.

[00:03:05] VC: Thank you for having me.

[00:03:07] JM: You joined Lyft in 2018. What were the most problematic infrastructure challenges when you started at the company?

[00:03:15] VC: I would say that – Well, I'm probably biased by my role, but I was hired to [inaudible 00:03:21] Kubernetes. At the time I would say the challenges came from that company having outgrown a lot of the infrastructure, like foundational pieces that they had. The company just grew very, very quickly. This is probably like a typical startup story, like they were super scrappy in the beginning. Just built whatever was like needed to ship the product and then sort of just like organically grew from there and just never had time to really like overhaul the systems.

It was sort of at a time where they were really outgrowing both in scale and also just like in the number of engineers that were like using the systems and it was like really painful and I can sort of like describe the stack a little bit if that's helpful.

[00:04:13] JM: Please.

[00:04:13] VC: Yeah. Pre- Kubernetes, Lyft ran on just like raw EC2 instances running Salt, and that was like great. It was like really fast iteration cycle and all the engineer had a lot of autonomy to configure the stack however they wanted to. This also sort of like supported our microservices architecture really well, and all the services used Envoy. I think the challenge came from sort of like having – By the time I joined, we had maybe like 300 or 400 microservices. You can sort of imagine like there a lot of different resources that we were creating on ABS and it was like becoming pretty hard to track and also really hard to sort of make sweeping changes to the entire fleet.

Another thing was we weren't using containers in production, and so it just was a lot of overhead in terms of like the amount of infrastructure and knowledge that an average backend engineer at Lyft needed to know to like do their job well. That was when we decided to introduce containers and also Kubernetes to sort of help abstract some of the more complicated lower-level stuff away and hopefully help the org move a little faster.

[00:05:30] JM: Had Lyft been watching the container orchestrator wars and just opting out until there was a winner?

[00:05:38] VC: Yeah, that is a great question. Yes. Lyft had been tracking the entire ecosystem pretty closely. When I joined, actually, I guess like a little bit before I joined. I guess I want to say when I joined, Kubernetes was may be like 1.7 or 1.9, something like that. Before that, like maybe a year or two before that, Lyft had – The engineering team at the time had made an effort to move to containers without Kubernetes. I don't know if like you remember, but like two years before that, there wasn't really a clear winner in the space. It was sort of like, Oh! You could use Mesos, or you could like can sit with this Kubernetes thing that was like sort of still a baby. There was also like Docker Swarm and a lot of other like players in that space.

I think a lot of what other companies started doing was like writing their own container orchestration layer on top of like Docker containers. Lyft did that that as well. What happened was like it was sort of like ruled out to development and testing, but it was never ruled out to production. Sort of like the team decided to wait for a more community supported platform to do real production rule out.

[00:06:55] JM: Right. This was the – To give your listeners context if they weren't developing backend infrastructure around that time. There were a lot of people who had built programmatic infrastructure with this generation of tools where you could script your infrastructure, the infrastructure as code tools, weather it was Chef, or Puppet, on in Lyft's case, Salt, and you could basically write code that would spin up your EC2 instances or whatever cloud instances you had. That was really great for a while until we started to realize the problems of this imperative coding for infrastructure. Then people decided that they wanted to have declarative structure or more declarative infrastructure, and that was around the time when there were all these container orchestrators that were open-source and some that were closed sourced that various companies were working on. It was a messy time, because it was a huge investment to go into one of these things until Kubernetes won and became a much easier choice.

Do you have a perspective on why Kubernetes won? Was it something superior in the infrastructure, in the architecture, in the marketing, in the evangelism? Why did Kubernetes win?

[00:08:16] VC: I want to say when I first looked into this space, it was probably 4 years ago, a little bit over 4 years ago. I was just starting at OpenAI at the time and we were building our infrastructure from scratch. We were like a new company. It was like blank slate and we looked at all the options that were on the table. We ultimately picked Kubernetes and we sort of like reevaluated every few months and we stuck with Kubernetes over my time there.

I think the reason why we did that and also probably resonates with other people who made the same choice is even though it's still like relatively young, I think the reasons why one has a very strong community backing, and you could call it marketing initially from Google or whatever. Ultimately, the community really, I guess, grew really strong over time. I think the architecture is also – I know a lot of people complain about like Kubernetes being like pretty complicated, but I would say for the amount of work that it does and for the functionality that it provides, the architecture is actually relatively straightforward to understand compared to a lot of the other orchestrators in this space at the time.

It was sort of like we had a very, very light infrastructure team. It was like two people, and Kubernetes was the one that we felt confident that with two people we could operate this thing and be in production and be okay. When we looked at like Mesos or some of the other things,

we thought that that was maybe a little bit too operationally intensive and like that we didn't feel quite as confident and like handling that with a small team.

[00:10:12] JM: How has Kubernetes changed the way that engineering works at Lyft?

[00:10:17] VC: I think it really – Well, one, I think it helped define sort of the interface between the infrastructure team and the service teams. I think prior to containers and Kubernetes, the line was like very fuzzy. All of our engineers could change the Salt code so they get could run whatever they wanted to on the EC2 instances. Obviously, there is like some amount of like base layer that the infrastructure team provides and it's like reviewed by the infrastructure team. But for the most part, they're like very empowered to own their infrastructure. I think now it's sort of like gives a clear interface between the two orgs.

I think another thing that serves like side benefit is like because it's open source, a lot of documentation can be Googled. The support burden is not just on the infrastructure team. Back when we had our own in-house stack, basically every question comes to the infrastructure team because we can't Google for like, "Oh! How do I run this command?" I think like it being an open source thing, one, like people can look for documentation, but secondly like their engineers that we hire from outside who have experience with Kubernetes, and so they don't need to be like, "I'm boarded on to this platform." They can just like start at Lyft and go.

I think the third thing is like – Maybe tying to like your earlier question of like why Kubernetes, is I think the Kubernetes API is very easy to work with and it's very good. It's like pretty straightforward to sort of like build our own tools on top of Kubernetes and make the tooling or the user experience better. Yeah, I think it's sort of spin up the infrastructure iteration cycle. Yeah, good common interface.

[00:12:15] JM: Can you describe how platform engineering teams and individual service teams work at Lyft and how they interact with each other?

[00:12:25] VC: Yeah. My team is, I guess, it's the platform team. We develop and manage and operate our Kubernetes platform and our service teams are – They own their own microservices, and so they own everything sort of like down to – I guess now they're on

Kubernetes. They own everything sort of like running in their container, and the interaction – Basically, I guess the agreement, is sort of like we provide a platform that confirms to Kubernetes and runs containers and everything below that layer, we own and manage, and we help them sort of have a platform that works out of the box for any Lyft conformant services.

I don't know if that answers your question.

[00:13:16] JM: I think it does. Are there instances where if – Let's say I'm a service team. I've built the –I don't know, the pricing service. My service calculates a price on a Lyft ride whenever a rider makes a request to Lyft. Are there days when I'm going to wake up and I'm going to come in the office and the infrastructure that I'm building on has been updated and improved by the platform infrastructure team or whatever the name of your team is?

[00:13:49] VC: Yeah. That does happen. For example, we have – Every service deployed on our Kubernetes clusters have a number of sidecars attached to them. There are number of ways in which the infrastructure can be updated without sort of any work from the service teams. One way is if the sidecars are updated. One of the sidecars is, for example, Envoy. If the networking team pushes a new version of the Envoy container, then all our services sort of like overtime get rolled over to the new version, which means that they get a new version of Envoy automatically. That doesn't require like any work from the service team and they might not even notice that's happened.

This also happens with sort of like underlying nodes. We update our Kubernetes versions pretty continuously to pick up any like patches and security updates, etc. We do roll our clusters and nodes almost weekly, and I think for the most part, the service owners won't notice. We train our nodes gradually and then they get scheduled on to new nodes and the old cluster slowly gets killed.

[SPONSOR MESSAGE]

[00:15:09] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you

can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:16:57] JM: You gave the example of Envoy. Envoy has been widely used in different circumstances as a "service mesh" or at least as the substrate for a service mesh. Can you tell me how Lyft has used Envoy? How extensively it is used beyond the initial use case of just being a service proxy? Do you use it for any of the "service mesh" purposes?

[00:17:29] VC: I guess we mostly use it as the service proxy. I think the way we use it is like all the inter-service communication goes through Envoy and we use it for like service discovery. Also, I guess all the traffic coming from external traffic endpoints gets into Envoy mesh and then gets proxy to our services.

I think one cool way that we've been using it is because we've used it before Kubernetes. We sort of leveraged it as a way to like seamlessly migrate some of our services on to Kubernetes without having –I think a lot of other companies sort of move on to Envoy as they move to Kubernetes. So they have to onboard, sort of like shift their traffic both into like the cluster and into the mesh. Because we were already running Envoy, we could just like very seamlessly ship to traffic in and out of our clusters and it just all appears as like normal Envoy traffic. I think I know a lot of the like newer maybe service mesh do sort of like maybe fancier functions. I don't know. Maybe that's how you're referring to, but I think we do use it for like – Basically, it runs Lyft. Yeah. We use it for everything.

[00:18:48] JM: You mentioned that a given service – So in this hypothetical pricing service, for example, there might be multiple sidecars attached to a service. How many company container sidecars might be attached to an average service? Can you tell me more about what those sidecars are doing?

[00:19:06] VC: Yeah. Depending on the service, we might have I think up to five sidecars attached to it. We have Envoy, which is running the proxy. We have a stat sidecar that's forwarding all the stats. I think we also have optionally log forwarding sidecar. I don't even remember some of the other sidecars. I think we have something like analytics sidecars for like events. But, yeah, the way that like services get deployed is that they can sort of like flag or opt out of sidecars, and otherwise we give them the default set of sidecars, which is like Envoy logs and stats.

[00:19:49] JM: What's the process of deploying a new service at Lyft?

[00:19:55] VC: We have sort of like these like standard service templates that help generate new services. Then I guess like the first step is you sort of like pick what stack your service is going to use. Are you going to be like a Python or Go service? Depending on that, you might have like different generators. Let's say I'm generating like a new Python service. There is like a template and like it gets automatically populated, and then maybe I'll like add some endpoints. Then I can just like deploy all our deployments, go through Slack. We just like go on Slack and sort of like deploy the service for the first time. Then the image gets generated, and we also

automatically generate the Kubernetes objects and then it gets deployed on to the cluster. This is sort of like a simplified version of that pipeline, but that's the gist.

[00:20:48] JM: Now that you've been at Lyft for two years, what are the canonical engineering challenges to the extent that you understand them? Are there some particular challenges that you think are characteristic of Lyft? Problems that will arise in different forms within the company for a long time?

[00:21:08] VC: That's a good question. I think one thing coming into Lyft that has been interesting is testing end-to-end has been interesting, because like Lyft is a pretty stateful service compared to a lot of other services. You can be in a ride and then the ride has like different states that it is in. For example, we've developed all these tools and services to help us test these things like to help simulate rides, for example, and like staging and in production so that we can always be testing all these different states that our clients could be in. I think because of that, it does present a lot of challenges when we're developing new features or like writing new services, is that just being aware that there are a lot of statefullness to the data.

[00:22:04] JM: Yeah. Do you mean statefullness in the sense that on an average ride, for example, you just have a user that is moving through the world physically in a ride and that ride has state? The user is accruing more expense over time or the ride is moving through space. Is that the kind of state that you're talking about?

[00:22:27] VC: Yeah. Basically, there is this like I see as a massive state machine that you're sort of like moving through different states. You're like requesting a ride and then it gets dispatched. Then maybe you get matched and then you're finally like moving in the ride with the driver. I think there are like a lot of edge cases to that. If the driver cancels, what happens? I think that does make sort of like testing trickier.

I think another thing is like we have all these – We're dealing with all the real-time updates from all the mobile apps. Dealing with sort of like low connectivity is another interesting problem. It's kind of like not in my team's domain or wheelhouse, but I think that's another interesting thing that we always test for.

[00:23:15] JM: Yeah. We did an Envoy mobile show with Matt where we touched on some of that. That does seem like another one of these canonical problems, is the interacting with the mobile infrastructure that is ephemerality.

Is Lyft still entirely on AWS or have you moved into other clouds or on-prem infrastructure?

[00:23:37] VC: I don't know that I can say specifically, but we're not exclusively on AWS. I think we're primarily an AWS shop, but we do have things running in other clouds. Well, yeah, and a little bit of like on-prem stuff.

[00:23:56] JM: Do you have any perspective for – Like, let's say hypothetically, I am building a big company on AWS and basically I want some minimum degree of failover into a multi-cloud scenario. What are the areas that are most critical for me to insulate? What should be my multi-cloud strategy?

[00:24:22] VC: That's a big question. I think the first thing to figure out is sort of like what you're going to do with your data. Are you going to like replicate the data across different clouds or are you going to like keep them in one place? If you, then you're like I guess not effectively like highly available. It's sort of like deciding if you want to be like active-active, or active-passive. Maybe the first step is like how much you want to be replicated.

Another thing is like sort of connectivity between the clouds. How crucial that is? Because there is a lot of like tricky parts like connecting things between the clouds. A lot of times you have to get like fiber from like telecom and then it's not like super straightforward and it can get pretty expensive. A lot of times it's not clear if like if the interconnect goes down, which end is down. There are a lot of questions that need to be fleshed out and also like how things fail and what happens when they fail. I guess like that's a pretty big and vague question.

[00:25:30] JM: I agree.

[00:25:32] VC: I don't know how deep to go into it.

[00:25:33] JM: Fair enough. Okay, more specific question. I don't know if you can answer this one. Do you use any cloud cost management systems?

[00:25:43] VC: We have our own in-house built one.

[00:25:45] JM: Wow!

[00:25:46] VC: Yeah. We do use – It's based on the data from AWS. Because we do attribution by like by team or by project, so we do have our own labeling on our jobs. Then our in-house system will scrape the data and it has like a nice dashboard and everything and we can sort of like see real-time how much each team or each project is using.

[00:26:12] JM: That whole space of cloud cost management, it's kind of interesting, because there are so many of these companies. I must've seen 10 or 20 cloud cost management companies at the various conferences I attended in the last year or two. Do you think this is – Is this an area like logging where there's just going to be multiple winners, or do you think that it's going to consolidate into somebody that just dominates at cloud cost management?

[00:26:41] VC: Yeah. Actually, that's a good question. I think I especially saw a lot this year at KubCon, I guess because Kubernetes gives people a nice API to sort of like scrape the data and give a nice cost dashboard. But I don't think there is like a clear winner. I'm not sure. I guess there is opportunity for there to be one.

Yeah, I think a lot of people I've talked to, they sort of like have their own in-house one just because of like there is a lot of specific data that like each org or each company wants right now. A lot of those other solutions are newer. I guess that's – I haven't looked too deeply into them, but I think some of their sort of like pricing models are interesting as well. Some of the ones I've seen are like some sort of like percentage of your cloud cost, which seems like it's maybe counterintuitive, because then our incentives are aligned. I want to save money and you want me to spend money.

[00:27:45] JM: True. You go to KubCon, and I go as well. One of the things I like about KubCon is that there is this meeting of the minds of different kinds of companies. For example, you have

people from Lyft mixing with people from an insurance company that's 100 years old that has layers and layers of legacy infrastructure because they're 100 years old. Lyft has its own legacy infrastructure because it's not the youngest company in the world anymore, but it's all post-cloud. What's your perspective on the Kubernetes related problems that these pre-cloud, these older enterprises are having compared to the Kubernetes problems that a company that is younger, a company like Lyft is having?

[00:28:34] VC: I think there's like a number of different ways depending I guess on what they came from. I would say one is like a lot of the Kubernetes materials or like conversations have been very – Well, cloud native, I guess. If the older enterprises have their own like data centers or colo, maybe that's a little bit less relevant. So they have their own set of challenges sort of like deploying Kubernetes on to their data center with like every static fleet. Then they also have questions about like whether they should move to the cloud. I think that's sort of like one big fundamental difference.

I think a lot of conversations, I guess, this year or like last year, have been on like development or like user experience. A lot of the conversations, I've been on like matching user experience, to like maybe what he were used to before, or just like making cloud native usable or user-friendly. I think maybe some of the workflows are different between like traditional enterprises and like more newer startups that have been on the cloud from like day one. I think that's maybe like another difference.

I think a third would maybe like less about the age of the company, but maybe more so about the type of company they are. I sort of see a lot of more traditional enterprises having businesses that need more security, and that's sort of like have been a big focus at KubCon last time, is like security or like regulation or governance clusters. I think that's another one that's jumping to mind right now.

[00:30:33] JM: Do you have a perspective for how the different cloud providers are diverging from one another or do you do you feel like they're just trying to maintain feature parry with one another when you look at AWS, Google, Azure?

[00:30:48] VC: Oh! Well, I think Azure has been sort of like leading in terms of like making everything super user-friendly and just like abstracting away everything that's underneath and just being like, “Okay. Here is the Kubernetes and this is what we provide and manage.”

I see AWS as like sort of trending that way, but like not quite as drastically or aggressively as Azure. I think Google has – They’re obviously like the most experienced in running Kubernetes, and I think they’re sort of maybe more in the middle, like not – You could use Google as like a sort of, “It’ll abstract everything,” sort of way, but like, again, not quite as extreme as Azure, I think.

[00:31:38] JM: What kinds of custom tools or platforms have you built on top of Kubernetes at Lyft?

[00:31:45] VC: I want to say like we’ve built a number of platforms on top of Kubernetes. I’m not sure if like that's what you’re asking about. For example, we have like our own machine learning platform running on top of Kubernetes that provides like interface for our machine learning scientists, and that's entirely sort of like running on Kubernetes and we have other sort of like develop for tooling I guess built on top of Kubernetes. I’m not sure if that's what you're referring to.

[00:32:16] JM: Yeah. Can you say more about that? Is it something similar to Kubflow or is it just completely homegrown?

[00:32:26] VC: Yeah, it’s completely homegrown. I think it started around the time when Kubflow was like – Maybe before Kubflow or when Kubflow was like very young. I'm sure we’re not the only company that has built a platform like this, but it’s sort of leveraging the flexibility and speed and scalability of Kubernetes to do machine learning training. It gives of pretty user-friendly interface for our researchers to like start any experiment they want to and that just bonds jobs on the cluster.

[00:32:57] JM: The auto scaling that you want out of machine learning training, that's what it makes simpler, because you need – Maybe you need to spin up a lot of servers to parallelize the training or something?

[00:33:12] VC: Yeah. I think there's that and there's like this flexibility that you get from using containers, because when you're training experiments, you're using very specific libraries to help you sort of define your model and your training. Rather than sort of like asking the infrastructure team to go install all these libraries to the machine and make sure it's all the right version and they're all compatible and everything, you can do that in your container image. I just tell the orchestrator to like go run this like 100 times or something. I think it empowers the users a lot more to sort of experiment and play with what jobs they want to run. Also, now they can sort of like scale easier, again, without the help of the infrastructure team.

[SPONSOR MESSAGE]

[00:34:10] JM: I love software architecture. Software architecture is the high-level perspective of how to build software systems. Much of Software Engineering Daily is about software architecture, and if you're interested in software architecture, there's no better place to go to discuss and learn about software architecture than the O'Reilly Software Architecture Conference, which is coming to New York February 23rd through 26th of 2020.

If you are interested in software architecture, you can go to oreillysacon.com/sedaily. That link is in the show notes, and you can get 20% off your ticket to the software architecture conference. The O'Reilly Software Architecture Conference is a great place to learn about the high-level perspectives and the implementation details of microservices, cloud computing, serverless and also systems like machine learning and analytics.

If you've been listening to Software Engineering Daily for a while, you know that these systems are hard to build and they take engineering details at both the high-level and at the low-level. Whether you're a seasoned architect or an engineer that is just curious about software architecture and maybe you want to become a software architect, you can check out the O'Reilly Software Architecture Conference at oreillysacon.com/sedaily. Use the discount code SE20 and get 20% off your ticket.

There are lots of reasons to go to the software architecture conference. There's networking opportunities. There are plenty of talks and training opportunities, and you can get 20% off by

going to oreillysacon.com/sedaily and entering discount code SE20. I've been going to O'Reilly conferences for years and I don't see myself stopping anytime soon, because they're just a great way to learn and meet people. So check it out, and thanks to O'Reilly for being a longtime sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:36:27] JM: Before you worked at Lyft, you worked at OpenAI. How do the machine learning workloads that you see compare between the two organizations?

[00:36:36] VC: Yeah. Well, OpenAi didn't have a production environment or like a traditional production environment. We weren't really like a service or we didn't have a product. I think that's one difference, is Lyft has to tackle a lot of the problems that come with model deployments and model version tracking. I think some of the things that are shared are sort of reproducibility of experiments is one thing that I think both organizations benefited from and switching to like containers and having immutable images and infrastructure. It makes experiments rerunning a lot easier.

I think OpenAI – OpenAI was a hybrid cloud environment. That's another big difference, is we were running Kubernetes across our colos and we were also running on all three clouds. It was just a much more varied environment.

[00:37:38] JM: The fact that you were not operating services that had to respond to user requests on an active basis at OpenAI, how does that change the nature of the company? Because something is different than – I can't think of another company, like another mainstream Silicon Valley company that has that feature. How does it change the nature of the organization?

[00:38:05] VC: Yeah. I mean, we could go really deep into this question alone. In terms of the infrastructure, there're a number of like differences, I think. One is – Well, maybe like this is sort of the experience running batch workloads, and a company like Lyft as well is – They're sort of more episodic, right? You sort of start your job or your experiment and then there's like a clear end to that job. It makes migration of infrastructure a little bit easier, in my opinion. It's still not

easy, but like just because like there's clear start and end, and when you start a new thing, you could just run it on the new infrastructure. It makes things a little bit easier and also makes reliability little easier to like sort of think about like I could have – I could take a cluster down and spend a new one up and just ask people to run their experiments on the new one. As far as like infrastructure goes, I think that's a little bit of like leeway that we have when we're running these like more batch type workloads.

[00:39:08] JM: Give me a few lessons in engineering management that you've learned at OpenAI and Lyft.

[00:39:14] VC: Let's see. I think the two companies are like pretty different. OpenAI was like a lot scrappier, and I think the – I guess like there's always this may be challenge when you're running infrastructure, is sort of like your team's relationship to the rest of the engineering organization, because you're sort of running a support organization and you want to make sure that the relationship between the two sites is healthy.

I've definitely seen infrastructure teams where like the relationship is like maybe like not so good or like even toxic. We were pretty mindful of that, and I think it's even harder in OpenAI to sort of like maintain that relationship, because not only were we supporting the rest of engineering, but there's also this question of like engineers supporting researchers and like there are these two different roles and they're like cross-functional, and it's like how to make sure that the two groups respect each other.

I guess it's not really like – There isn't really like a silver bullet, but just some things that I found that helped build empathy between the two groups is to set up regular communication or forms between the two groups to like let people talk face-to-face. I think a lot of times when people only interact with your infrastructure team through like support channels, it can feel very impersonal. They're only talking on Slack or something, and a lot of sort of like emotions or like tones get lost. Sort of encouraging a lot of these like in-person interactions really helped.

Another thing is like especially with the researcher and engineer divide is having people shadow each other or like even just embed in each other's teams really helped with that, like empathy building. I think a lot of engineers didn't understand the complaints that researchers had until

they like actually sat with a researcher and understood like, “Oh! This is why the infrastructure is painful.” Those are I guess tips that I learned.

[00:41:28] JM: I don't know how much you have looked at the state of academic research or corporate research. Do you have any perspective on whether OpenAI is perhaps maybe a structure or a model that other domains could follow? Because I think there – I just hear from scientists that are at least in academia that are a little bit frustrated with the incentive structures and they feel pretty hemmed in. Although I realize OpenAI is a very specific kind of organization. So perhaps it's not the most replicable thing. But I don’t know if you have any thoughts on that area.

[00:42:07] VC: Yeah. I’m not sure that I know enough about academia to comment on it. I do think that I guess like OpenAI’s structure is very good for like experimenting with maybe riskier ideas or giving people a little bit more freedom to experiment with many different research directions.

[00:42:30] JM: Totally separate question. Do you have any – I know that you joined Lyft in 2018, and so I imagined the firefighting was much less than some of the earlier stories I've heard. But do you have any good firefighting stories from your time at Lyft or experience responding to tough incidents?

[00:42:51] VC: Oh, actually, yeah. I think this is a pretty interesting one, because actually we had similar experiences at OpenAI. Our machine learning cluster, we sort of like implicitly – Well, we used to anyway, implicitly trust our users to start their experiments without a lot of constraints on like how much they could use.

I think there was one time when – So the way that a lot of them start experiments are like they have a Python notebook or like a Jupyter Notebook and they sort of like explore the data from there. Then once they've decided on like, okay, what features they’re going to use or what type of model they’re going to use, then they like start their experiments and they just like start the experiments directly from the notebook.

They may be like write some for Lyft and live through some of the data and launch experiments that way. But there was one time I think sort of like late at night when one of our researchers wrote a loop and then they went home. It ended up being like an infinite crash loop. Eventually, it's sort of like it paged the machine learning team first and they like wasn't sure what was happening and they like couldn't access the cluster anymore, and they paged my team and we were like, "Oh! That's weird." The control plane got overwhelmed and went down and sort of like that was the initial symptom that we saw. We just like brought the control plane back up and we're like, "Hmm. It just seems like etcd got like overwhelmed. Let's just bring it back up."

As soon as it came back up like within a couple minutes, it went down again, and we didn't – I think our initial suspicion didn't go to just like something within the cluster was like spamming the cluster. Yeah. After we brought the control plane backup, like another time we were like, "Okay. This is like – Something's fishy here." Then we started looking at objects that were like in Kubernetes and then we realized the pattern and then we found the job that was spawning all these things. Yeah, then we killed it and it was fine, but it was like pretty funny because for a while like teams are like, "Are we being like DOSed? What's happening?"

[00:45:16] JM: One of the episodes that we did with Matt Klein from Lyft was when he talked about a problem of human scalability, which is this idea that hyperscale companies can sometimes have very difficult operational problems that cannot necessarily be solved with technology. Have you encountered any of these at Lyft?

[00:45:40] VC: Yeah. I think there's a lot of sort of operational or like support that our infrastructure team does, and maybe that overlaps with like some of what Matt was referring to, is like as we've grow, the infrastructure team has to support an entire like engineering organization, and over time that's sort of like adds up time on the team. It's like when somebody comes and asks like about their job failing or the deployment failing, like stuff like that, little bit of time here and there adds up. Then also sort of like operating a cluster that's running 400 or 500 microservices. That's a lot sort of like broader of a scope than what's used to be.

[00:46:26] JM: Last question. How do you anticipate the Kubernetes ecosystem changing in the next couple years?

[00:46:31] VC: Yeah. I think maybe this is like a little bit of a personal wish as well, is I think the ecosystem is maturing. I think a lot of the basic functionality is there and the pain points I see people having right now is like, “Okay. Now that we are running Kubernetes in production, how are people developing on Kubernetes?” That’s sort of like story that's not super fleshed out yet, is like what is the user experience? It’s like pretty steep learning curve right now and it's not super integrated into people's like development tooling. I think that's going to be an area that needs to be answered. I think other sort of like polished things are, yeah, security and governance. Those are all things that as more mature companies are adapting the platform, they’ll need to figure out solutions for.

[00:47:28] JM: Well, Vicki Cheung, thank you for coming on Software Engineering Daily. It’s been really fun talking to you.

[00:47:32] VC: Yeah, thank you.

[END OF INTERVIEW]

[00:47:42] JM: As a programmer, you think an object. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers and the cloud area. Millions of developers use MongoDB to power the world's most innovative products and services, from crypto currency, to online gaming, IoT and more. Try Mongo DB today with Atlas, the global cloud database service that runs on AWS, Azure and Google Cloud. Configure, deploy and connect to your database in just a few minutes. Check it out at mongodb.com/atlas. That's mongodb.com/atlas.

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

[END]