EPISODE 1531

[EPISODE]

[00:00:05] LA: As applications grow in size and complexity, and as they increasingly move to microservice architectures, it becomes harder for individual developers to perform end-to-end tests of an entire application stack. Connecting development services to production services is off limits. Test environments are limited to automated tests. Staging environments are woefully inadequate with the amount of data available for their use. Testing large and complex applications is becoming harder and harder. Speedscale provides developers a solution to this problem. Speedscale assists in developing and testing applications by recreating real-world traffic loads, and test and development environments. Essentially, bringing the data quality of production into the exploratory world of development.

Matt LeRay is the CTO and co-founder of Speedscale, and he's my guest today. Matt, welcome to Software Engineering Daily.

[00:01:09] ML: Thanks, Lee. It's great to be here.

[00:01:11] LA: Glad to have you here. I've been talking to Nate, one of your co-founders as well too. He talks a lot about the value of Speedscale for deployment, testing, and using production data for that sort of testing environment. But Speedscale has a huge benefit in helping developers build and test individual services that are part of a larger – microservices that are part of a larger application. But the developer experience, can you tell me how Speedscale helps developers during the development process?

[00:01:47] ML: Yeah. First off, great introduction on Speedscale, Lee. I think you do a better job of explaining it than we do. So that's great. Yeah. When you come to the developer desktop, one of the big challenges as an engineer is always getting, getting a set of test data that mimics production. What we do is, if you kind of think about writing unit tests, we go through and we take our best guess as engineers. We say, let's go and write some unit test that simulates the number of permutations that we can fit into our unit test or our imaginations. With Speedscale, just to back up a second, Speedscale is essentially a window into the production system. We replicate the data. You can think of it almost like a developer preview environment, or an ephemeral environment, but pre-populated with all of your data.

So instead of having to replicate every database in the production system, Speedscale takes a copy of the traffic that it sees, and then pretends to be those production systems. It's kind of like a very clever way of mimicking what's there without actually have to set up the infrastructure.

If you think about that, in the context of this new wave of cloud native development, you get to a point where you can copy out what's happening in production, and create a full integration test environment on your desktop. Let me kind of go through a few things that we do with that, even in our own developments. Naturally, our software never breaks. I've never written a bug. I've heard of other people that have. But me personally, no bugs, obviously. But when I'm debugging someone else's bugs –

[00:03:33] LA: They're just new features, right? That's what they are. They're not bugs, they're just unknown features, right?

[00:03:37] ML: Yeah. I mean, I intended for it to do that. The screen should be blank right there. One of the things like, let's say, we're debugging an application. There's kind of two big use cases. The first one is, I am going to first off try to figure out what the heck happened in production that caused this thing to break, caused my service to break, what input, what API call, which exact set of query parameters, which exact set of whatever it was caused my software to break. The first thing Speedscale does is, we allow you to view all the traffic going into your service, and then selectively pick out the transactions that broke them, and rerun them like a curl command against your local debugger, or any environment you want, even another Kubernetes cluster, QA environment, ephemeral test environment, et cetera. We allow you to go just clip out those transactions and replay them, so that you can watch it happen in real time.

[00:04:35] LA: This is out of a service-by-service basis, you can do this?

[00:04:38] ML: That's right. Yeah. In any service in your environment. When I say environment, Speedscale, we started with the new wave stuff, which is Kubernetes. But we also work in pretty much anything you can imagine like Beanstalk, or Docker, or virtual machines, or just pretty much, you name it. We have some sort of support for it. We'll go and clip that out and bring it back to developer desktop.

[00:05:02] LA: Cool.

[00:05:03] ML: That's kind of use case one, is reproducing those issues. Use case two is actually reproducing the environment itself on the fly without configuration. Now, you've got your – let's say you're running it in a debugger even. You're going to running your service. We can go and replay the traffic. But then, we can also clip out all the transactions that were going to downstream systems. Let's say you've got one of those death stars, like Netflix, where you got a thousand different microservices all talking to each other. We will go and say, "Okay. During this time period, this service talks to five other microservices, here's what they talked about and we'll go and replicate that on the **[inaudible 00:05:43]** desktop. When the same code that you're testing out in your debugger makes those calls, Speedscale will be ready and waiting to hand back the correct responses, so it thinks it's in a real production system. Those are kind of the two big things that we do on the developer desktop.

[00:05:59] LA: Does it to that with a script, so it says, "I'm expecting this from the service. Now, I'll send this as a response and expecting this, sending that"? Or does it do it as a, like a state machine saying, "Well, if I get this, I'll send this back in. But if I get this other thing, I'll send it back. You see the different things I'm saying there? I mean, can you make changes to your service without just completely redoing the script if the order of calls to service change or something like that happens?

[00:06:30] ML: That's a good question. The system, we call it a responder, but the service mocking engine, our responder goes and it runs like a state machine, and that has a pre planned set of responses. It's based on the idea of full automation. We sort of subscribe to this Kubernetes idea of, we don't want to curate individual scripts, we don't want to write scripts. We want to treat it more like something that we recorded or you programmed, like through API calls. But it can kind of be blown away and rebuilt from scratch. To answer your question directly, we start from a preset, premade set of responses like a state machine, but then it's programmable in the sense of you can say, "Okay. I have this new set of responses I want, "Okay, let's make an API call." Let's go and change the responses, let's add some here, let's nip talk. Let's curate the responses we want. It's not a script, but it allows you a lot of the same flexibility, but it acts more like editing a JSON file.

[00:07:32] LA: Got it. Essentially, you record from production together and the – I hate the word script here, but I'll use the word script, you just put in air quotes. You record what happened in production to get the script of what went on. That becomes the starting point for the development environment on your desktop for testing the services, state responses that go with it. But you can adjust that, and tweak that, and tune that and program it so it does different things at different times, if necessary. Am I saying that correctly?

[00:08:08] ML: I think you're saying it perfectly. I'll just add a little bit onto it. Recording and production is something that's extremely hard and very valuable, so we talk about it a lot. But you know, there's a lot easier things we do as well. For instance, if you'd like to start from a Postman collection, because a lot of us develop against the Postman collection, or even a swagger spec or whatever. We'll go import that Postman collection and then we'll use that as the starting point. Or if you want to read a log file, as long as it's sort of some sort of structured log, we'll read does. So we talked about recording and production, but it's really anything you want to give us, we can consume, and create the starting point.

[00:08:42] LA: Got it. Cool. Essentially, we're saying import this. What you're importing is you're creating a JSON file that describes this whole process. It's your own internal programming language, if you will, for these scripted mocks or whatever you want to call them, and it's a JSON file. Is that correct?

[00:09:01] ML: That's right. In some cases, gigantic JSON file. The only difference, I guess, I'm remembering my computer science classes from college now. What we do is – it's declarative, I think. Anyway, it's not really a programming language. It's not like a step-by-step set of instructions. It's more like a declaration as opposed to being a script that you're going to mess with. But yeah, you got the idea.

[00:09:25] LA: Yeah. I always use the word script in quotes, but it's a descriptive word, but it's not a technically accurate word. I do get that. Yeah. But now, the JSON itself, you can edit the JSON directly, but you provide tools to edit it as well, right? UI based tools and things like that. Is that correct? Or how does that work?

[00:09:47] ML: So if you've ever used a log viewer, one of the real sophisticated logging tools, like a Datadog, or an Axiom, or Elasticsearch or any of that. You will recognize our interface, because the things that we know about, the responses and the transactions we know about, we put it into a webinar face like that. You can go navigate around. It looks a lot like Postman, which is actually a partner of ours as well. So we'll go through that. When you want to make the edit, edits, you can either do it like through API calls, or like command, like a command line tool, or you can actually go and directly edit the JSON file. We build all this fancy stuff, to give people the ability to click around, and move this around and do stuff. What we found at the end is people actually still love SED. SED, the Linux command to do finds and replaces. So you want to go across multiple gigs and gigs of transactions, you can do that. It's actually really easy to do. It's just a big file.

[00:10:43] LA: We are talking software developers here, so they like text files, they like command line, and yeah. I say they, I really mean we. I know what that's like. It's amazing. I completely avoid text files and command lines for everything. I do this on programming. But once I start switching back into developer mode, I'm right there again. Everything has to be a command line. You know how all that works. Go ahead.

[00:11:15] ML: No, I was going to say, it's interesting. When I first started – well, not first started, but when I was building software in college, the vi and Emacs was the big debate. But it was the best tools we had. It's just what existed. It's funny, as we bring on more and more engineers at Speedscale, it seems that that is what everybody's going back to. Or not everybody, but many people have gone back to on the backend side. One of the things we measure when we're going through designs on our software is, do we have to take our hands off the keyboard and touch the mouse or not? Because that is like a key element of developer happiness is if you have to take your hand off the keyboard, then you're going to start irritating engineers now. Don’t touch the mouse.

[00:11:54] LA: Yeah. That makes perfect sense. In some ways, building user interfaces for developers is easier than it is for any other class of user because of that. I mean, you know exactly what they want, you know what they want, or what they want as a simple hands on. They don't mind complex command sequences. They just want to be able to do everything they want to be able to do quickly and easily, and it doesn't matter if it's complex or not. But it's a lot harder when you're trying to deal with making user interfaces for non-developers, right? Because it's there's a whole different set of expectations for them. In some ways, developers are an easier demographic to be working for. Along those lines, your primary user is a developer. But is that the only user or are there other classes of users you have and you work with your product?

[00:12:55] ML: Yeah. Developers, so one of the observations I've made over the last few years is that, engineering leaders don't want to invest in specialized QA teams anymore, because the perception is that, having QA slows you down. That this kind of – it became an obsession, like how many releases are you doing per day, per month, and everybody was sitting around, kind of comparing that for a long time. I think it drove everybody to say, you know what, we can't really have separate QA testers. I think this has actually contributed to developers being unhappy. Because writing tests is actually really hard, and figuring out how to break software is the opposite of trying to make it work. It's like a different mental model. So, yeah, a lot of the folks we interact with day to day are developers. I would say, there's also a class of larger organization that still must have QA, because of various reasons, right? They must have QA. They're kind of evolving into what's called an SDET or software development engineer and test. There's sort of an evolution towards SDETs or just eliminating QA altogether.

I'd say, those are kind of our two sets of folks that we work with the most is engineers and SDETs. What we don't do is, we eliminate a lot of the script writing and the complexity around maintaining integration test environments. We just get rid of that stuff, because we're coming at it from a completely different angle. We don't interact as much with just like pure traditional QA engineers that are doing manual testing, because what we do is a lot more – it's almost like too simple. Does that make sense?

[00:14:38] LA: Yeah, it does. But definitely, it's the hands-on developer or hands-on test, depending on the organization. That's the primary user. They're not managers. It's not architects. It's not higher up in the organization. It is the rank-and-file developers that are primarily using your product.

[00:15:03] ML: Yes. One of our screener questions, whenever we would talk to folks used to be, "When was the last time you use VI or Visual Studio, and when was the last time you ran kubectl or kube CTL or whatever you want to call it, the Kubernetes command line interface. There's a battle around that. But those are the people that – those are our people, right? Anybody who is cranking out code, trying to accelerate delivery and hates writing tests, but it's touching a keyboard directly, is our kind of folks.

[00:15:31] LA: Right. Makes sense. If you had to train, generally classify your application into a higher-level classification, what I come up with is something like production API simulator. But that's a pretty limiting definition. Can you give a better definition of your category of where your product fits into the ecosystem?

[00:15:59] ML: Yeah. This is actually – this is a complicated question in some ways, because the type of category we fit into exists at some of your very, very cloud for companies. If you go through and look at companies like Facebook, or Netflix or what do you call it? DoorDash or Google, right? You will find in those companies that as they moved on to more of a cloud native architecture, they tried software development, they tried to do pure test and prod. Meaning less, only test and production. Then as they sort of created bigger and bigger blast radius, they found it very hard to control, letting their customers test their software. They eventually end up building something like Speedscale. So you can actually look through a lot of blogs, and I did before we started this company. You can see where those companies all essentially evolved into needing a Speedscale, like automated mocking and testing tool that is based upon real user interactions and real microservice behavior.

That class of system, we're calling it production simulation engine. You might come say something slightly different, that it is actually a –it is ephemeral – it creates developer ephemeral environments in the CICD pipeline, but that actually replicates data. We don't actually compete with a lot of the tools to try to replicate your entire production system, because that's more of like a deployment problem where they have to copy every database system and every yamo. What we do is actually a lot simpler, because we just do a little micro environment, surrounding your service where we emulate everything it needs. We don't emulate the entire environment.

So we're not really, surely like a preview or a femoral environment thing. But we kind of solve some of that. I think that we're in a situation where, as the market emerges, as developers learn this sort of new way of just in time development. We'll get a better definition as time goes by and we'll get a clearer market definition. But of course, we're still pretty early days with those things. We're pretty confident in the approach, but we're going to change the naming around as we move along.

[00:18:12] LA: Yeah, that makes sense. You're creating your own category here, or at least, you along with a few of the more forward leaning companies in this space. You're creating a new category here. Let's talk about the different areas where Speedscale helps. We talked about the developer desktop environment, where they're building or debugging a service and using you as a simulator for everything else. There's that environment. There's integration testing, that you can be used for. There's a service development integration testing, there's pre-deployment validation, you're running a series of test based on scripted or mocked. I should say, real user data before you go into production. What about things like at scale stress testing? Will you help in those sorts of situations? Or is that really a stretch for what you're trying to accomplish?

[00:19:20] ML: Yeah, that type of testing is dead in our strike zone. So stress testing comes in a lot of different formats. So one of the simpler forms of stress testing is doing an HPA for Kubernetes, figuring out how to size your cluster for various – or how to tune your Cluster Autoscaler. These are things that are pretty typical uses of Speedscale. So they just map that out for everybody who doesn't live this world every day. What will happen is, let's say like you're a big streaming company, you use the Speedscale recorder and it is a very high-volume recorder. It can grab from lots of different pods at once and pull it all together into one big snapshot. So you're going – you do this huge recording, and you'll say, "Okay. This is an hour of traffic or half an hour." Then, we'll go through and say, Speedscale has the ability to make 100 users seem like 10,000 users by suddenly modifying the data.

The simple version of this is changing usernames, but it's a lot more complicated than that. We'll change unique IDs, and we'll kind of zip through and move all the data around. We'll take that thousand users and make it seem like 10,000, let's say. Then we'll go and they'll plug in Speedscale, and do run after run in various environments, and then compare the performance of those environments using Speedscale, because we produce reports with all the usual graphs. That will help them sort of tune their Autoscaler.

That's kind of actually an interesting first use case for a lot of folks here as they move to the cloud, is they say – they moved to Kubernetes at least is, how much do I need? How big can my cluster be? How many nodes should I allow to scale up with, not bankrupt myself? So Speedscale will give them sort of a realistic sense of how big things need to get right out of the box.

[00:21:05] LA: It's not just stress testing. It's the sizing. It's giving sizing information about how to size your production environment.

[00:21:13] ML: Yeah, that's right. Yeah. I kind of expanded the definition there. Yeah.

[00:21:16] LA: That's cool. You mentioned you caught yourself at one point and you actually hit something I want to talk about. You said, cloud switch to Kubernetes. Now, I imagine the vast majority of your customers are Kubernetes users, because they're microservices, they're interconnected microservices. Kubernetes is a great environment for that. But what about cloud? Are all of your customers cloud based? Or are some of them on-premise or hybrid customers?

[00:21:53] ML: It's actually splitting up to be more about half and half now? Kubernetes, if you listen to – I don't know, the latest Dora report from Google. But Kubernetes is, actually, I don't want to misquote them, but it's something like, half of all large enterprises are using it. It's more than half. I have to go look up the exact statistic. But it's an enormous amount. That's true. But what we see actually very commonly is there's a lot of companies that are moving from different kinds of cloud native architectures to Kubernetes. What that caused us to do at Speedscale is just equally support all the environments that we encounter, because the technology doesn't really change much for us. It just gets more or less seamless.

If you're in full Kubernetes, all you got to do is run one annotation and Speedscale does all its magic, like it's fully automated. If you go to something like EC2, you're going to go in the AWS console, and you're going to set a few things up to reroute the networking or whatever. It's not hard, but it's stuff that needs to be done. It's actually about half and half between a variety of different environments. One of the things we're seeing actually too is the rise of things like OpenShift. Is it OpenShift? The Red Hat flavor of Kubernetes? Is that right? We're seeing a rise of that, because there are folks who are wanting to run Kubernetes, but they want to run it in a slightly different way and customize it.

Another one, we're seeing more, Spectra Cloud, things like that. But anyway, I think we're living – we're going to be for the foreseeable future, living in a polyglot technology landscape, where Kubernetes is going to take over big chunks of the workload, as people move over and try to take advantage of it. But then they'll also learn the limitations of Kubernetes, the areas where it's not as strong. You may remember this, but running databases in Kubernetes used to be just an exercise and pain. Still kind of is, but it's getting better and better. There's still a lot of workloads that you still need to keep out of Kubernetes in one way or another. There are cloud services. So yeah, we've kind of fully embraced that. But when you start a company, like Speedscale, and we start out with nothing, what we're looking for first and foremost is to find folks who are doing the very cutting edge, and we go and make them extremely happy. We want to make those people super happy.

Then, as you get bigger and bigger, and you pull on more customers, then you start expanding the aperture a little bit. So we've done the same thing here, so we've kind of moved out of Kubernetes into things like GCP. Google is expanding quite a bit. We have customers running in Azure. Think about Kubernetes, right? They're just running VMs and windows servers and all that stuff, so we're kind of all over the place. You'll see all of it.

[00:24:40] LA: It's Kubernetes and other cloud environments. But what about on-premise environments, are you finding any customers use you in an on-premise environment?

[00:24:49] ML: Yeah. We have one customer I can think of off the top of my head that is using us in Bare Metal VMware. They have their own servers and they've got – I think vSphere armor, which version but they're running vSphere on there. Then they have us running alongside their services and recording. It's a very, very large tech company that isn't going to modernize on a dime. They've got a lot of variety stuff. That's fine with us. VMware is good. It works great for them, so we're happy to do it.

[00:25:20] LA: But for the most part, your customers, if they're not migrated to Kubernetes, or not migrated to cloud environments, they're in the process of doing that. They're trying to modernize and using you as a tool to modernize. You're not seeing customers who are staying in the old way of doing things adopting you. You're finding people who are adopting you are the ones who are trying to modernize the move to modern infrastructure. Is that a fair statement?

[00:25:47] ML: I think that is – yeah, I think that's fair. They have to have a mindset that they would like to automate these things. Usually, when you have an automation mindset, you're going to adopt cloud philosophies, which means you're going to be going that direction.

[00:26:02] LA: Yeah, you're right. That's almost an assumption, right? Why would you be automating if you're not trying to try and do –

[00:26:10] ML: Well, I will say. I'll do one carve out to what you said, though is, when you get into things like healthcare and financial services, these are companies that it's not about modernization. They can be hypermodern. I don’t know if you've ever seen some of the trading platforms. I've worked with a number of them. We have a big one here. Actually, in Atlanta called Intercontinental Exchange. There's New York Stock Exchange. There are all these different trading systems that over my career, I've talked to and these folks have hypermodern environment. They care about sub millisecond transaction times, like just the fastest stuff in the world. But they can't go to a public cloud because of regulatory concerns or other issues. We're happy to meet those folks there as well. But to your point, Lee, I wouldn't call it a niche, because there's a lot of money made through financial systems. But it's not what we hear about the most, those kinds of systems.

[00:26:58] LA: Yeah, it's always funny, whenever I find I do the same sort of thing, or when we're talking, except for a couple of key industries. And those key industries that are always healthcare, or some form of finance or private banking, something in that realm. Those two areas seem to always come up.

[00:27:15] ML: You know, all those people with all that money. Yeah, those folks.

[00:27:18] LA: Right, exactly. All, but about 50% of the business, but that's – This is great. How did Speedscale start? What's the origin of the idea? You're one of the co-founders. It was the three of you, right? It was you, Nate and Ken Ahrens, right?

[00:27:43] ML: Yeah. Ken Ahrens, Nate Lee and me, Matt LeRay. I was actually coming back from California due to some family issues. My wife, she is from Georgia, and her family's here. I won't get into details, but she wanted to kind of be closer to family through this next stretch. I was thinking, okay, what I want to do next? A lot of my background was in monitoring. I'll tell the story from my perspective. Monitoring is a very mature space, right? Observability is what we call it today, which is an expansion of monitoring.

[00:28:18] LA: We've both been in that space for a long time.

[00:28:21] ML: Exactly. You know exactly what I'm talking about. You have some really great companies on there, the Datadogs and New Relics. My last company, you got all kinds of folks working on that space. My kind of view was that, as a startup, starting outside of the Bay Area, I felt like that area is well served, right? There's a lot of companies that are thinking hard on that and innovating there. What can we do to kind of turn the industry on its head a little bit in our little corner of it. I started talking to Nate and Ken, and both of them had worked at this really cool company called ITKO way back when. ITKO was purchased by a big company called CA, where it was kind of retasked towards older technology. It didn't – it's doing its thing or whatever, right? But the idea was basically, let's not build service mocks. Let's go and generate them from other data sources. Now, ITKO was – it was limited by its era. In the era of ITKO, they had a lot of great ideas, but you couldn't just do something in Kubernetes, where you reroute the networking. That wasn't really possible.

Most protocols, like wire protocols, networking protocols were closed source, Oracle databases, things like that. You couldn't really decode things with high fidelity. But if you fast forward to 2020, when we started Speedscale, most protocols that are heavily in use are going to either be some sort of variant of HTTP and JSON, or they're going to have an open-source library that you can go and use to decode them. The recording problem, couple that with like Kubernetes dynamic software defined networking, it's a lot easier. Then we kind of said, okay, my background, I had worked with a cloud data warehousing company. I kind of had gotten an idea of that new approach to handle massive datasets. I said, "Wait a minute." The rules around big data, like how big data systems are built are like totally different than they were five years ago, the compute costs, the ingest costs, everything is upside down." It used to be really expensive. Now, it's really cheap."

Then, we kind of put those two things together and said, let's go and let's take a crack at this. We started the company in 2020, early 2020. We went through Y Combinator in their first remote batch. We actually went through their startup school, because – if you ever going to start a company, do it with people that you love to argue with, because you're going to argue a lot. You must as well love doing it, right? Do what you love. We said, "Okay. If we're going to argue about all this, let's argue with Y Combinator instead of each other."

We signed up for this thing called, it's called startup school online, which I highly recommend to any other startup founder. They go and they teach you all the ropes of how a company is built, what are the basic things. You don't really want to innovate on things like setting up an S corp or something. These are all just basic, legal things. Y Combinator walk you through this. As part of that, it turned out that we were kind of highly ranked amongst our peers, and Y Combinator asked us if we wanted to apply, we did. We were fortunate enough to get in and have some great mentors. Went through that, and then we started collecting some more big enterprise customers, figuring out exactly what they needed, how this thing should be shaped. We kind of went through 2020, and piece of 2021 figuring that out. Then, we brought it to market and the rest is sort of history.

[00:31:45] LA: Cool. Where are you based right now?

[00:31:50] ML: Our core team is in Atlanta, Georgia. Before that, I was just south of San Francisco, right before the pandemic hit.

[00:32:00] LA: Cool. What's next for Speedscale?

[00:32:03] ML: So glad you asked. We seem to have figured out the shape of our product somewhat. It's helpful to people in – so what people do is they'll take Speedscale, they put it into their CICD pipeline to do continuous validation. They'll do things like stress testing, like nightly builds. They'll do stress testing against it. They'll have individual – every time an MR is built, they'll go and run like common scenarios recorded from production against that. We're adding more and more customers doing that.

But the more exciting thing for this audience is, we are getting ready to release a free version of our product that is perpetually free. What we really want to do is take some of the learnings that we've, that we've paid for in blood, frankly over the last couple of years, figuring all this stuff out. We want to make it accessible to everybody who is just trying to dip their toes into it. We're figuring out the shape of that free product right now. So hopefully, we can start to contribute back and help people with their day to day. And of course, we are running a company. As things grow, we naturally want them to come to us as their needs grow, as they get more teams involved, et cetera. But we want to make it easy for people to just get value out of what we've learned, without having to make a big investment. That's kind of the next big thing for us is a "free tier" as an –

[00:33:36] LA: If people want to find out more about you, what do they do?

[00:33:40] ML: The easiest thing to do is go to apeedscale.com. You can also find us on LinkedIn, and Twitter and all those places. If you really want to learn about it, we would love to learn from you as well. What I would do is go to slack.speedscale.com and join our Slack community. Then, you can go and talk to the founders directly and tell us what we missed, tell us what ideas you have and we would love to connect and sort of develop the space together.

[00:34:10] LA: Our guest today was Matt, LeRay, the CTO and co-founder of Speedscale. Matt, thank you for being my guest on Software Engineering Daily.

[00:34:18] ML: Thanks for having me, Lee.

[END]