EPISODE 1139

[INTRODUCTION]

[00:00:00] JM: Developer tooling and infrastructure is a fruitful area for investing. A wide variety of technologies can have large investment outcomes based on the fact that there are lots of engineers and businesses that are willing to pay for products that give those engineers a higher degree of leverage.

Lee Edwards is a partner with Root Ventures. His focus on hard problems within software makes him a great guest for the show. And he also talks about the thesis for his firm as well as his personal beliefs on what makes a good investment.

[INTERVIEW]

[00:00:36] JM: Lee, welcome to the show.

[00:00:38] LE: Thank you. Yeah. Thanks for having me.

[00:00:39] JM: Every venture capitalist has their thesis for how they invest. Tell me a bit about your thesis.

[00:00:44] LE: Our thesis broadly at Root is what we call hardtack. So we're interested in things that are technically challenging that require engineers on the founding team that are may be like roadmap-driven. And so that spans I think from like AI and ML to software infrastructure, robotics, heart industry. I think almost anything that has like a physical hardware component kind of counts as technically difficult if you've ever done it before. The other piece of it is that we're all engineers in the areas that we invest in. So my area is pure software, which is kind of why I've been spending a lot of time looking at like developer tools and cloud services for developers and applications of AI and ML, things like that.

[00:01:23] JM: Could you give a few examples of investments you've made?

[00:01:26] LE: Yeah. The one that we just closed last week, I don't know that I can talk about yet. But it's in the DevOps space. One before that is called Meroxa, Meroxa.IO. Essentially, the CEO, DeVaris, he was a director of developer experience at Heroku. So it's very kind of like user-friendly sort of way to spin up change data capture. Sort of run a single command line command and connect multiple database sources to data destinations and have it spin up Kafka and set up sort of a subscribe, or the published part of the pub/sub architecture. And then kind of expanding from there into like what other kinds of things can be data sources, like third-party APIs and what other kinds of things can be data syncs, like Slack, or maybe PagerDuty, or something.

HASH.ai, which is doing a modeling and simulation web-based framework. So you can think of it as kind of like Glitch for building complex like agent simulations. Joel Sapolsky is the chairman of that company. So there's a lot of I think Glitch inspiration in the UI. And Superconductive is a data quality tool you may have seen great expectations. It's the open source projects that it's built around. But it's all about like let's write unit tests for our data. Let's sort of sort of apply kind of – I don't know if I should say Agile or Mature. But let's say like let's bring some more like sort of software engineering quality principles to data and data management.

[00:02:51] JM: Tell me how one or all of those companies is representative of trends that you're seeing.

[00:02:58] LE: Yeah. I think they all are, and maybe in different ways. For HASH, I think it's really cool, because it's sort of – I think modeling and simulation is kind of an overlooked area, and I think the kinds of computer science that are being brought from academia and the industry. If you look at like the academic interest in – If you kind of Google agent-based modeling or complexity theory and Google Scholar, you see just like a huge uptick in academic interest in this. But I think it's not super, super popular in the industry yet. And I think one reason for that is it's kind of like building a game engine. Like you're sort of building your own run loop send. It's kind of a weird way to think about programming.

I think, yeah, games are maybe the best analogy. So I think that's a really exciting trend that's just sort of like not on a lot of people's radars, and we kind of expect that as the tooling gets better and if sort of make it accessible to all kinds of software engineers and data scientists and

analysts, that it'll be much easier to sort of solve problems with this tool.

With Superconductive, I tend to think of just like – I'm reminded of kind of the way that – If you kind of remember like the way it felt when, say – I wasn't around when like Agile was first coming around. But let's say in the early 2000's. It was like really, really exploding and we were seeing it enter all kinds of companies. The energy behind that movement and just sort of the inevitability of like we're going to figure out how to do things better. Like this pain is here. And all the things that we were doing at the time, like test-driven development, behavior-driven development, getting like really intentionally, like reading books about how to do re-factoring. That kind of stuff I think was like super exciting. And the culture that that was created by a lot of the people writing that stuff, and it was a lot of names to credit. Kent Beck is one I've been kind of rereading, revisiting lately. I think that will happen to data.

I think there are there a lot of folks that have sort of entered a complacency of like, "Well, data cleaning is hard. Data quality is hard. It's 80% of my job. I have a PhD in this field, but 80% of my times is data cleaning. And, oh well, that's just kind of how it is." And I feel like that there should be a way to solve that problem and let data scientists spend 90% of their time on data science.

And then with Meroxa, I'm really excited about just the idea of like I would love to just talk to everyone who is doing – Who has a developer experience like DX background and is trying to start a company. So I think it's a super interesting idea that like some people, they look at something like that and they're like, "Well, Kafka exits. Confluent exists." But I think the history of software engineering really is this kind of like building abstractions on top of abstractions. And I thin, really, I mean, my strong opinion here is there's definitely going to be people like sending me DMs. Like I think there is sort of an old guard way being like, "Well, if you don't understand all the way down the stack, you'll never understand it. You'll never be good."

And I think, inevitably, those people – Everyone has a limit to some point when they stop understanding the stack. Like unless you can explain to me like the Maxwell's equations that govern a transistor. At some point, we all kind of lose the abstraction layer. And so I think it's interesting to see like – Yeah, like there's going to be a lot of people that have data. They want to build a streaming architecture. They want to add some pub/sub to their application. And even

sort of like manage Kafka is a bit like, say, maybe even a day or a two-day project. And then for every data source and data sync that you're adding, even more so. Yeah. So I love the idea of like let's just figure out how to make this a single command line, a single command, and just sort of make this more accessible to even someone with like, say, fresh out of boot camp or fresh out of college.

[00:06:33] JM: So you are particularly focused on core software processes and software companies. Are there opportunities outside of pure software that are interesting to you or interesting to Root VC more generally?

[00:06:46] LE: Oh, yes. So many. I mean, I think outside of the stuff that I'm investing in, but still inside of Root. Like super excited about a lot of automation and robotics that's around different kinds of industries that had been experiencing labor shortages. And I think trends towards on shoring and near shoring and just kind of looking at the macroeconomic trends of labor productivity in certain industries kind of slowing.

I think there is a ton of stuff there, and we've got investments in construction tech. Let's say like Crux OCM – Sorry. Crux OCM is in oil and gas. We've also invested in Dusty Robotics, and Versatile, and Construction. So I love that kind of stuff.

And then even outside of that, there is a lot of stuff that I love and follow. I mean, I used to spend a lot of time in e-commerce and marketplaces and tools for creators. But I think our idea is to be a little bit more disciplined. Like I don't just sort of invest in everything that I think is super cool and exciting at Root. We have a very – We believe that sort of having hands-on experience building things like this kind of will make us better investors. And that may or may not be true, but that's what we believe.

I have actually a lot of angel investments are in stuff like that that I was just really passionate about. I have a few investments in sort of tech for women, which I think is like a really interesting trend at the moment, women starting companies, building products that were typically made by men before. So you see some of that DTC. And then Gypsy Stories is one that I invested that's doing sort of audio romance content for women. So they're sort of trying to take like the digital erotic experience for women and have it be like women-made and for women. It's a little bit

unclear why for so long that hasn't really been the case. Yeah. When we say we focus on hard tech, it's not just because that's the only thing I think is going to grow in the future. It's just that I think we're trying to be very disciplined.

[00:08:33] JM: Getting back to the engineering side of things, are there any other trends in developer tooling that you're seeking companies in?

[00:08:42] LE: Yeah. There's definitely a lot of stuff. I mean, I was tweeting the other day about game dev. And I don't know that necessarily we end up doing an investment in gave dev. It's possible. But I think I look at that and feel like there's a lot of opportunity. I think, in particular, things like Unity. I think I would even probably say like Roblox kind of belongs in this category. But things that sort of make it easier and easier to make games, but you still need sort of like studios. You need like multiple people of creative disciplines. There's even just like so many modelers and artists and shaders and designers and story borders of like just the amount of stuff involved in making a game that looks good.

And there're a lot of like really interesting trends. I think I called out Rosebud.AI, which is Lisha Li's company. And also Runway.ML, which are both kind of oriented towards a lot of like generative adversarial networks we've seen are so good at creating human-looking video that we have to worry about what happens if a digital president of a country declares war. It's like that good. So clearly, it should be good enough for games.

Yeah. I'm definitely interested in that area. Just sort of how do you – Given that games is the largest area of the mobile app store, and it's larger than the movie industry. Where is all the tooling? Where's the ability for like just two people to make like a blockbuster game?

Yeah. I mean, I'm always interested in looking at things that kind of – That sort of like add more collaboration. And this is going to be like a broad area, right? But I think just looking at sort of the workflow of developers and sort of what makes developers productive. I think it's often more sort of like more people in process than anything else.

Tools, they kind of like reduce that feedback loop. Just make a tighter and tighter loop between I think of what the thing should do and then the thing does the thing. And so I think, yeah, a lot of

what we've seen in developer tools has – GitHub is the extremely obvious example. But I think, anything, I would even consider things like CircleCI. Things that sort of sit somewhere in the developer flow and make the developer more productive.

[00:10:49] JM: And the stage that you're investing in, what are the economics like for a seed stage investor?

[00:10:55] LE: Yeah. What's my business model?

[00:10:57] JM: I guess like how big of an outcome do you need to make the investments worthwhile?

[00:11:02] LE: Yeah. The typical rule of thumb – It's really fun. When I first started getting into investing and before I joined Root, I was kind of poking around and trying to explore it. Could I start my own microfund? For a lot of reasons, I'm really glad that that never panned-out. Right now would be pretty difficult time to be doing that on my own.

But I put together like Monte Carlo simulations for – Based on some data that I was pulling out of like Crunchbase and trying to figure out, yeah, what are like sort of the range of outcomes for a venture portfolio. And the math that I came up with was basically what everyone says all over the Internet, which is for a venture fund to kind of get its promised 2 or 3X return for its investors, you need to have a deal that returns the fund. Meaning, you have $100 million fund. You need like a $1 billion exit where you own 10%. And that seems to be the case historically for funds that are working. Even funds that do well that don't have, you always got this like storied outlier of like the Uber or Zoom seed investment or something, right?

There are a lot of funds that have succeeded in 3, to 5, to 6X returns that like don't have the generationally large exit. To be clear, they've all got the billion-dollar outcomes. But they don't all have the 100 billion-dollar outcomes. And so that's the case if you're smaller fund, because if you're Softbank, returning the funds means an enormous number. You've got a few billion under management.

We have – This fund is 76 million. The previous was 31. And so one reason that we sort of want

to invest early and sort of work really closely with a founder, like often take a board seat or at least a board observer seat, or at least be like we want to be like their best and most active partner, like the sort of silent cofounder is a phrase we use. So if we do that and we sort of have meaningful ownership in the company, we can have an exit that's maybe even slightly less than a billion and still return the fund.

It's venture capital. Like the numbers need to be big. You sort of can return a fund on a bunch of like small acquisitions. But if you read Peter Thiel's *Zero to One*, like a big thing he talks about is he's like, "These markets have to be in the trillions." He wants to find the next Facebook. Not even the next PagerDuty would not be like the win for a founders' fund, because it wouldn't return the fund. But it would be a huge one for us.

[00:13:25] JM: How does doing investing compare to building products and writing code? Doing engineering work? For those people who are out there and considering a career in venture capital.

[00:13:35] LE: Yeah, it's pretty different. And I think, for me, it was like maybe a little bit of an easier – It's still a weird transition, but a little bit easier given a couple things. Like one I still code all the time. So I kind of don't – I'm certainly not doing it as much as I used to. But I'm doing it enough that it's fun and I look forward to it and I stay up-to-date. And so I don't really feel like I'm not building anymore. I certainly feel like I'm building less and it's not the number one focus. But I don't feel like programming is like a past tense activity for me. So that's definitely kept me happy, because I've always loved programming.

The other thing is kind of moving from like being an engineer to being a manager, to maybe being like a manager or managers, to being like an executive, like a C-level person who's sort of involved in company strategies and going to board meetings. You sort of are already on this path of like pretty common saying of sort of like moving from control to influence. By the time you're sort of like managing managers or like managing people who manage managers, you don't get to it and you don't want to tell people what to do. Instead you're trying to do a lot more guidance. You're trying to create an environment that gives you the outcomes that you want. And I think occasionally maybe drilling down and sort of getting your hands dirty possibly from time to time. But for the most part, it's like sort of an influence job.

So I was sort of already on that path by the time I jumped into venture, and I did spend a year full-time angel investing. I mean, I didn't have a job. That was my job. And I did that at least in part to figure out if it was something I would like. So I think like what's really different about it that I didn't expect is – So, certainly, if you're not someone – If you leave those days where you're like, “Wow! That day sucked. I had six hours of meetings.” Boy! Venture capital is not the right thing for you.

I think it's also like very much not a 9 to 6 job. I think maybe for some VCs, it is. I think there are a lot of VCs who sort of made their money and made their names. And maybe they kind of live on the beach and they attach their name to the masthead of the firm and they raise capital for it. And that's possible. I think there’s also probably a lot of – There may be a number of VCs who don’t take the job super seriously and may be, are there for like the networking drinks, which at this point hopefully it's on something else.

But I think if you're new and you’re trying to make a name for yourself and you’re trying to work your way up, it's actually pretty important that, at least in my mind, like I know that the founders I work with are not really working 9 to 6 and if they need to call me at 9 PM or 10 PM. I took a phone call at midnight on my birthday in another continent. And I'm actually pretty glad I took the phone call. It was pretty important. Kind of like personal issue that the founder needed help with. And that's just kind of how it is. I think you sort of – There are so many great things about it. I think it's the best job I've ever had. I think it suits me better than anything I’ve ever done and makes me so happy. But it kind of only works I think, if you can kind of handle those things. You needs to be able to have hours and hours and hours and hours of personal meetings or sort of one-on-one meetings rather. And I think you need to be like extremely flexible. Your work-life balance needs to be more of a work-life smoothie.

[00:16:50] JM: Coming back to the side of things of developer tooling, how have you seen the go-to-market strategy for developer tools change over time?

[00:17:00] LE: Yeah. I mean, if you were to go way, way back. Look at like Borland C++ or something, right? There was a time when developer tool is a big-box software that you’re literally buying a box and stick the discs into the computer. So to be honest, I don’t even – Yeah,

know that much about the economics of those things other than it did exist. All the way to I think open source is clearly sort of become one of the biggest ways that a lot of developer tools do distribution. So Nadia Eghbal has a really good book about this that just came out, and I'd be shocked if I was your first guest that name-checked it. Have you read this? It's Nadia Eghbal's book, it's called *Working In Public*.

[00:17:38] JM: I haven't read that yet. It's on my list.

[00:17:41] LE: Yeah. It's amazing. So I think it's pretty clear that if you want to get your developer tool into as many hands as possible, making it free, making it – Lowering the barriers to entry, making kind of the hello world really great. Actually, it's kind of funny. I think a lot of companies we work with actually have kind of invented or are using, maybe they're borrowing, I don't know. But they're using terms to mean the same thing, and they all have their own term who say like the magic first-minute or the awesome first 10 minutes is what Superconductive says. And I think the magic first-minute is what one of our other companies says. But the idea of just sort of like we've all seen this, right? It's like the 15-minute how to build a blog and Rails demo. Would Rails have taken off without that video? Or if it had been like 30 minutes, if it had been something un-relatable, like not a blog. Maybe style, the woops style kind of made it name worthy.

Yeah, I think the big thing that's happened here, and it happened in IT, and it's also happened in like HR, enterprise software to some extent. Or less so HR, but in other kinds of enterprise software to some extent, is people kind of bring their tools to work. And that's especially true to developers. I think a lot of times, the fact that developers often have development as a hobby at home means they'll try something out. And when they fall in love with it, they're going think of it when it comes up at work. Or they may even just like pound the door down and basically insist that work needs to be used. I think Slack benefited from that a lot. I think certainly GitLab and GitHub as well.

So yeah, I think, open sources – This is maybe like one main thing that may be further outside observers, and maybe like some VCs don't get, is like open source is not really a thing about labor. It's not about like you get a lot of engineers working on the thing for free. It's about things like NPL. Like package managers and source code repositories are just amazing free

distribution. So I think the biggest companies and dev tools look like that.

There are certainly plenty of very valuable companies that are doing top-down enterprise sales approach. So I was at Pivotal back when we were a software consultancy. And then eventually bought by EMC and sort of – You probably are familiar with like Pivotal Cloud Foundry, this kind of stuff. VMware has bought them now. So they are a very top-down. Like if you look at their – With the time that they IPO'd, they had like three digits of customers. I can't remember how many it was. I want to say maybe 300 total customers. But the average contract value was huge. I think it was in the millions.

I mean, that can exist. That's still a real thing. But I think if you're trying to look for sort of like what's the next company worth in the tens of billions, I think it has that kind of consumer-looking growth trajectory, the sort of viral word-of-mouth.

[00:20:23] JM: There is a gradient between the hosting providers and the low-code applications. And I see this gradient as kind of the developer-based platforms, like Render and Netlify, or getting higher and higher level. And the low-code applications or no-code applications are getting lower and lower level. Do you think there's so a merger, an overlap coming between those two? Or are we already there?

[00:20:50] LE: Yeah. I think that's really interesting. I tend to think of the low-code, no-code kind of concept that has gotten a lot of buzz lately especially like among VCs. I tend to think this is a little bit more of an evolutionary rather than revolutionary change, to be honest. I feel like I hope I don't like discourage people from reaching out to me if anything I'm minimizing the thing that they're working on.

Like I said before, like it's definitely just always been the case, that software is about abstraction on abstraction on abstraction. And I know a lot of really great developers who built really cool things and work at like really amazing companies and are like they have a lot of experience. They've very senior. They mentor people. Like awesome developer. Who like has literally never compiled C code before.

So I think that a lot of these tools are moving us in that direction. And I think, yeah, maybe no-

code's mission is a little different, which is just sort of let's make it so anyone can sort of build applications. But as I kind of said in the sort of tweet storm that got me on the show, I think that we really underestimate who can write code and what they can do with it.

If you look at people that are writing formulas in Excel or people that are writing like HTML, or CSS, or even like markup, markdown for content, there are a lot of people that go to school with an English degree and leave knowing how to use HTML. And I think that there is a sense that like the hardest part of solving any problem is understanding the problem defining the problem. Having the background and the expertise and then like doing the thing. And if part of your solution is to kind of like Google around and hit up Stack Overflow and then like edit a line of Python, I think a lot of people can do that. And actually a lot of people are doing that.

There are ton of people that are like, "Oh, I don't really know how to program." But they go into the edit pane on WordPress and they change a few things. So I think that no-code is very important, but I kind of am more excited about even what's no-code with a little bit of code, which I guess at that point maybe you just call that code. I don't know.

So I'm very, very excited about things like Netlify and Render. I think they're super cool. And I almost think of those as almost like no DevOps, or at least let's say less DevOps. I think there probably always will be DevOps. But hopefully like the things that we're spending our time dealing with, kind of the mundane things, will sort of go away more and more. We're pretty rarely like shelling into machines and like uploading private keys anymore. There's a lot of stuff like that that tooling has just sort of abstracted from us in the DevOps world. And I think Netlify does a great job of that, so does Render. And we have a couple investments in this space as I mentioned, at least in the space of like removing DevOps.

[00:23:34] JM: And what else? What are the kinds of tooling has been abstracted away by no ops kind of stuff?

[00:23:40] LE: Yeah. I think, yeah, Docker, Kubernetes is an obvious thing. But I think one thing that's kind of funny about that is, in some ways, I actually think those obstructions are like still relatively immature relative to where they're going to go just because I think if you look at something like Kubernetes, it actually does take away so much complexity and replaces it with a

lot of other complexity. And I think that that's fine and good. Kubernetes is still like maybe not necessarily the first thing that you want to configure when you do a Hello World. And I think there's just sort of a lot of apps that don't like need it necessarily.

But, yeah, I think I've seen so many people working on different ways to make that easier. But I kind of think – And like without giving away sort of our most recent investment, like one reason I'm excited about it is I kind of think a lot of the big applications of Kubernetes and sort of zero DevOps is like what can you build on top of it?

So I think it's possible that the best Kubernetes startups, so to speak, that we see right now are not necessarily going to be we help you orchestrate your Kubernetes. But more about we hope you solve this problem. And by the way, we're using Kubernetes, which makes it way easier for us to solve your problem.

So yeah, I think that's a huge trend. I mean, things like Snowflake I think are super interesting. I don't know that you necessarily call this DevOps. I think you probably do. I think there's a good amount of plug-ins and tooling and data infrastructure around Snowflake that makes it just kind of a lot easier to maintain a more complex data warehouse.

I mean, Heroku, is kind of the obvious one that people talk about. And Heroku is, I think, these days – It's owned by Ssalesforce. I think Salesforce's roadmap for Heroku is very clearly like not for us. It's like they're definitely kind of working on supporting more of the Salesforce ecosystem.

I was talking to Jason Warner recently, who is CTO at GitHub, and he was previously – I believe he was CTO at Heroku. He's like, "Yeah, the crazy thing about Heroku is I think it actually created like way more value than it captured. If you look at the influence that Heroku had on the world and on the ecosystem, and on the way developer tools are made, and the way that developer tools are documented, and the way that we think about how we should be interacting with our hosting environments, created the entire like platform as a service revolution, or evolution maybe, either way, that the concept. It's kind of mind blowing.

And he didn't say this. But to me, it's almost a shame that Heroku is not a like $10 million public company, that actually they could have been maybe the company that invented Netlify, or sort of

formalized the Jamstack pattern and made a bunch of offerings. But they weren't. But I think we owe so much to that point of view. The idea that developers should just provision their own machines and not worry about it, and it's totally okay if you don't really know how to shell into a box and like install some flavor of Linux and get the package manager running and make sure all the dependencies are working and document like what versions all the dependencies ought to be. That wasn't even that long ago. I'm not even old, and that's how I got started programming.

[00:26:51] JM: What about the spaces where – There was this meme probably – I don't know, last couple years that you can't invest in companies that AWS is going to eat. And we've kind of seem time and time again that actually you can, and whether it's Confluent, or MongoDB, or Redis Labs, or Elastic. All these companies can be successful despite the fact that AWS has like the Amazon basics version of your – Kind of the knockoff version of your product. Are there any things that AWS can just obsolesce by cloning the product? Or have we learned that that's just not the case?

[00:27:29] LE: Yeah. I love that you asked it in that direction, because I feel like so often it's the other way where people are just kind of assuming Amazon will just crush anything you do. So like – Man! I could talk about this forever. So there're so many points here. Okay. So first of all, like AWS is not even a monopoly in cloud hosting anymore, right? It's exceeded its monopoly and is now in a three-player. You might call it an oligopoly, but it seems to me to be a race to the bottom in terms of pricing and sort of like a commoditization of everything that they offer. And may be the way they differentiate is brand. Amazon is kind of like, "We're the best. We're the biggest. We're the most reliable. And Google is kind of like, "We're the machine learning people." And Microsoft is kind of like, "If you like Microsoft, do the Microsoft thing."

So it's a pretty telltale sign that they're offering a commodity. They're all trying to differentiate on brand with the offerings very quickly converging. When Amazon copies peoples open source projects and offers their – And I'm going to start using that AWS basics. I mean, like "Are using AWS basics Kubernetes? So they can't steal the community, right? And that's where the power is as we talked about before, right? The growth comes from that community. That's what makes the sort of venture scale network effects. And so far, there is no community that's been like, "Oh! Well, I guess we'll just kind of go work on the Amazon version of this."

And then the other thing is I think a lot of VCs in particular, they don't notice – If you don't load the AWS panel from time to time, you don't realize how many products there are. They fail all the time. There is like hundreds. There're so many things on the Amazon. I can't even use AWS anymore. The primary mode of interaction with AWS control panels is a search bar. There're so many products. Most of them fail. So there's plenty of room.

And then the other thing is multi-cloud, right? So like, sure, Amazon SageMaker is a thing. But it works only for Amazon. And people are already multi-cloud. As you mentioned, I think there are few people who are 100% on a cloud provider. Once they get to a certain level of complexity, maybe their data warehouse is in Snowflake. Maybe they're using PagerDuty for alerting. And there are even some products that will be outside of AWS as cloud always, likes Stripe, for example, or maybe Twilio, or Planet Labs, or Orbital Insights, or all kinds of third-party APIs.

So I think at this point is pretty easy to have a lot of really important business logic and a huge important part of your application outside of a major cloud provider. And then I think at that point, that's – Sorry to like keep kind of selling my book, I guess, but it's what I spend my time thinking about. But it's like what's kind of exciting to me about Meroxa is like even if you're not on multiple cloud hosted providers, you certainly do have sources outside of the AWS cloud. So you could certainly configure Kafka and sort of creative a pub/sub ecosystem and do some change data capture that way. But so much of your important stuff is coming even outside of developer tools, right? Like we have important data that's coming in from like HR systems where like you're building a lot of internal applications at a company. Like data coming in and out Zenefits, or like ADP, or like Lever for your applicant tracking software. That data can be more and more important. So I just don't think AWS has a monopoly on this, like they never will.

[00:30:51] JM: Given that we are talking my cloud providers, could you give me your competitive breakdown for AWS, Google Cloud and Azure?

[00:31:00] LE: Yeah. I'm like certainly not an expert on this. So I think you probably sort of need to be in the weeds and shopping around the cloud providers to be. But yeah, as I said, I think that like AWS tends to be the sort of like buy IBM option. Google, because of their relationship with TensorFlow, I think, and also the custom hardware that they're building that's related to

TensorFlow, I think they really want to be the cloud for people doing machine learning. And I think they've been successful in that.

Azure, I think at first was very much sort of this works best in the Microsoft ecosystem. But what's been kind of interesting and like just really fascinating to watch at Microsoft is how they have really expanded their ecosystem. And it's kind of .NET is not really like the most important feature of the Microsoft developer ecosystem anymore. I think part of that is obviously GitHub, which I think may be one of the greatest acquisitions of probably like top five. It's nowhere near WhatsApp or Instagram. But it might be one of the most – Or DoubleClick. But it it's probably one of the five most important acquisitions in kind of the GAFA-FANG universe.

So what Microsoft has been doing that's really interesting is using the GitHub brand and the GitHub umbrella to do a lot more and reach out to more kinds of developers. So I would say it's becoming less and less true that Azure's biggest value prop is Microsoft. I mean, Microsoft is also looking to own the code editor here, right? I've never thought in a million years that I would be using the VS Code. I've been using them since I was like 22. And I switched to VS Code like six months ago, and I love it. I really wish I didn't, but I do. I use them key bindings, obviously. But other than that, honestly, I think that they're converging. This was sort of the suite of tools around SageMaker that AWS was doing. I think maybe they've got, as I said, a few different kinds of angles where like Google maybe has a lot more hardware and a lot more close access to the TensorFlow team. Although, to be fair, TensorFlow doesn't even have a monopoly on model architecture anymore, and Amazon – One thing that Amazon did that I thought was really interesting was shipping some like developer kits with hardware. But they had this DeepLens kit that was computer vision. Sort of Hello World for building a computer vision application on the Amazon Cloud.

I don't know if that really took off, but it's such an interesting idea. It's just like one of the ways that Amazon could really verticalize their competency in kind of building first-party consumer products and consumer electronics, and like selling and distributing them online, and then using that as lead gen to get people on to the cloud. If they could pull that off, that feels powerful.

And then Microsoft, obviously on the other side, right? If you're kind of not going to avoid using GitHub, like maybe there are folks using GitLab. But I think every person who uses GitLab but

work for other probably uses GitHub at home. And even if you're using GitLab at work, they're probably using a ton of packages that they're getting from GitHub native open source projects. So that maybe Microsoft's kind of acquisition channel.

[00:34:08] JM: Since we're on the topic of cloud providers still, what do you think of layer 2 cloud providers? We've talked about Render and Netlify. Are there any other opportunities?

[00:34:17] LE: Yeah, I do think so. It's a tough space for me to think about, right? Because I actually haven't really seen the pitch for like we're competing with Netlify. We're the new Jamstack thing. I haven't seen that pitch. I think it would be really difficult right now. It's very possible, right? I think Netlify as well as it's doing is not – They're not a $10 billion company that's completely un-seedable. But it does seem hard. It seems like really, really hard. They've done such a good job. So I don't know. I guess I don't spend a ton of time thinking about how to get into that space. But I do think that Jamstack is very powerful. I think that the tool chain is a lot harder than a lot of people think. If you're not sort of familiar with the crazy world of JavaScript, and Node, and VS6, and Typescript, and multiple package managers, like it's a crazy world. I think if you're kind of used to it, you maybe forget how insane it is. But even just stepping away from JavaScript for maybe a year, a year and a half and jumping into my first like Netlify, Jamstack project a couple months ago, I was just like, "I don't know anything." I can even write like a Hello World right now. I don't even know whether I'm supposed to use import or require.

I think that all of that could be cleaned up. And I don't really know how you do it, but I think that that's maybe one of the biggest things for growth of Jamstack. And I think that there's probably more than one, but not like two dozen sort of winners in the Jamstack space. I think there could maybe be one or two hosts versus, all right, it's another I think pretty popular Jamstack host. I think there could probably be some set of like services, sort of like what are the – Yeah. If you just think of sort of like all of the like LauchDarklies, or CircleCIs, or PagerDuties, or these kinds of things that kind of attach to the developer workflow. Like maybe there are things like that that are for Jamstack. It's possible. It's also possible that Netlify and the other tier-2s like own that stuff as just features and tabs inside of their dashboard.

I think it's a crazy interesting space, right? Once you kind of wrap your head around how to build

these applications, it's kind of like obvious. Like if your application is conducive to a Jamstack approach, why would you not do it? It's just kind of like the perfect architecture. And it's funny, like so many times – I think basically everyone has probably built a Jamstack application before the term Jamstack was a word. You would just call it like static assets served from the CDN. We did that a Groupon. Everyone does that at scale. It's just that your tool chain of like compiling the stuff that goes on to the CDN and like handling the caching and everything, that part is super hard, and the tier-2 providers kind of handle it for you. I think everything that's old is new again is pretty true in the developer tools world.

[00:37:10] JM: Well, Lee, that seems like a good place to close off. Thanks for coming on the show. It's been real pleasure talking to you.

[00:37:14] LE: Yeah. Thanks so much. It was great.

[END]