EPISODE 316

[INTRODUCTION]

[0:00:00.4] JM: Making the right engineering choices in today's wide landscape of cloud technologies is hard. Predicting the future in order to invest in companies in this space has the same level of complexity. The cost of cloud computing is going down, but the volume of total required space and processing power is going up. The open source community is growing and improving, but people are increasingly willing to buy software that will save them time.

There are many countervailing trends. Capital is flowing into Silicon Valley at a faster rate that could be sensibly absorbed by the number of quality companies that exist. A decent product won't have trouble raising money, but a decent investor has to choose wisely among the huge selection of available opportunities. Lenny Pruss works at Amplify Partners where he is currently focused on the cloud native space.

In this episode we talk about what cloud native means and now to navigate the complex landscape whether you are an engineer or an investor. I enjoyed this conversation a lot and I want to thank Tom Tunguz for introducing me to Lenny. Tom's episode was a while ago. We talked about his book; *Winning with Data*. If you like this episode, I encourage you to check out that one.

[SPONSOR MESSAGE]

[0:01:25.4] JM: Life is too short to have a job that you don't enjoy. If you don't like your job, go to hired.com/sedaily. Hired makes finding a new job enjoyable, and Hired will connect you with a talent advocate that will walk you through the process of finding a better job. It's like a personal concierge for finding a job.

Maybe you want more flexible hours, or more money, or remote work. Maybe you want to work at Facebook, or Uber, or Stripe, or some of the other top companies that are desperately looking for engineers on Hired. You deserve a job that you enjoy, because you're someone who spends

their spare time listening to a software engineering podcast. Clearly, you're passionate about software. It's definitely possible to find a job that you enjoy.

Check out hired.com/sedaily to get a special offer for Software Engineering Daily listeners. A $1,000 signing bonus from Hired when you find that great job that gives you respect and salary that you deserve as a great engineer. I love Hired because it puts more power in the hands of engineers. Go to hired.com/sedaily to get advantage of that special offer. Thanks to Hired for being a continued longtime sponsor of Software Engineering Daily.

[INTERVIEW]

[0:02:51.4] JM: Lenny Pruss is an investor with Amplify Partners. Lenny, welcome to Software Engineering Daily.

[0:02:55.6] LP: Thanks so much for having me, Jeff.

[0:02:57.3] JM: You've been studying the cloud native space and writing about it, trying to figure out what investments to make, what technologies are, getting traction. What does the term "cloud native" mean?

[0:03:10.2] LP: Yeah. There're a couple of things to unpack. I think starting at the highest level, cloud native is really an architectural paradigm, a software architectural paradigm that describes that how apps are built as resilient distributed systems that scale to meet the demands of millions, or tens of millions of users. This paradigm was really sort of informed and inspired by warehouse scale computing that takes place inside of organizations like Facebook and Google.

Then, when you think about what a cloud native app is, there's sort of this technical definition that people have logged on to where it's microservices oriented. Microservices, being that these applications are composed of sort of discreet loosely coupled services. These applications that are dynamically scheduled and orchestrated, and then the container is the functional unit. That's sort of the technical definition.

Then, there's this notion of a cloud native stack that we've been delving into, and that's really sort of the infrastructure and tools that go into building, deploying, and managing these applications.

[0:04:15.2] JM: One way that you framed this cloud native architecture is in terms of these layers. Maybe you could describe the layers of the cloud native architecture.

[0:04:25.5] LP: Sure thing. So when we talk about the layers, we're really talking about all the components that go into running a production grade distributed system. When you're running an application that lives as a monolithic code base on a single server that's one thing, but running decentralized apps that you're trying to scale to millions of end users, that brings about an unprecedented levels of complexity.

So the tools needed to build these apps and the components in the stack needed to support them are different. Just a few years ago nobody really talked about things like service discovery or scheduling. If anything, these were elements confined to the world of high performance parallel computing. Now these are household tools that are part of everyone's lexicon, anyone who's trying to build microservices.

To provide a little back story, when Docker came out in 2013 and as its market started to mature, I started mapping all of these projects and all these companies to try to make some order out of it. So I put a bunch of these projects in a stack to try to visualize how everything fit together and kept tabs on this ecosystem. Then last year we teamed up with the Cloud Native Computing Foundation and the Technical Oversight Committee, which at the time itself was voting on the CNCF reference architecture.

The goal of this was to really provide a more definitive taxonomy and reference design that would detail many of the prevailing projects and companies that make up this ecosystem and make some order out of the chaos, just to give folks who are entering the ecosystem or practitioners who are building these systems, a better sense of how everything mapped together.

Ultimately, that work was represented by the Cloud Native Landscape, which was a document that we released at last years [inaudible] Con in Seattle.

Now to get back to the layers, again, these are loose definitions and all of this relates to elements that are needed to run a distributed systems. But if we want to walk through the stack, at the bottom is the infrastructural layer and obviously this relates to the physical and/or virtual compute network and storage resources that serve as the building blocks for any application. Moving up, we've got the provisioning layer and this represents sort of the tools needed to stand up and manage those infrastructure resources. These are tools that are things like Terraform, Puppet, Chef, and also things that are necessary to secure this infrastructure.

Moving up, there's the runtime layer, and that's where the container engine, most notably, Docker, lives and that's responsible for operating system level virtualization. In this layer you tend to include also things like cloud native storage and network resources that interface with the container engine and extend down into the infrastructural layer. Then you get into the orchestration layer, which is really the management plain. Here you have tools like Kubernetes, Docker Swarm, Mesos, as well as elements like service discovery and service management.

Most basically, this layer really plays the role of a conductor in an orchestra. It takes all the underlying resources and services and composes them into a logical application and serves that to your end users. Finally, above that is what we call the application definition and deployment layer. These are really a bevy of tools or a bunch of different tools that developers use to build their apps and then to deploy them. So things like languages and frameworks, CICD platforms, source code management and the like.

The important thing to note about all of this is that this strives to be a Canonical, if not overly simplistic representation. In real life, elements of all these layers extend up and down and at the end of the day, this is really sort of a three dimensional, living, breathing organism.

[0:08:09.2] JM: Absolutely. I think a lot of this has basically been driven by this movement towards cloud infrastructure and the shift towards cloud infrastructure was such a fundamental change that it led to all of these other advancements that improve the efficiency of our stack, ultimately. They improved he leverage that an individual programmer has. The cost in some

ways is the complexity. Although, I guess, there was just perhaps equal complexity in an on premise setup 20 years ago, but that complexity manifested in different ways. Now, the complexity manifested in ways like we've got so many layers like you just described. You've also got the observability question. You have an entire side of your application devoted to understanding what is going on in your application. There are — Billion products are on this. I'm sure we'll get into that.

What are the canonical problems that developers are dealing with in the cloud native ecosystem? Obviously, one way to look at all of these things is that they are creating opportunities for more leverage. Another way of looking at it is — Especially from an investment point of view, is as we get these increasing layers that are adding leverage, there are also new problems that crop up from an investor point of view. I'm sure some of those look like opportunities. What are those Canonical problems?

[0:09:31.3] LP: Of course, any new platform shift, a customer's problems or pain points, are really a start up's opportunities. The way I've seen this ecosystem evolve, Docker came along and provided this really neat lightweight abstraction for a developer to take their application, wrap it up with all of its dependencies, and be able to deploy it on any Linux based server. So you had developers everywhere using this cool new tool called Docker as a universal package manager.

Then they started to deploy Docker and then there were all these coordination challenges around service registration, discovery, networking, orchestration. Really around challenges of now that you've got your app, or these independent little microservices, how do you manage them at scale? How do you manage them across hosts? How do you do networking across hosts? The canonical problems then were as simple as, "How do I make a couple of these services running on two different nodes where they talk to each other and turn those two services into a logical app?"

More complex issues were around, "How do I manage state in this environment? What does networking look like?" Again, between hosts or between data centers. So what you had was a bunch of projects and companies that formed to help solve around some of these Day 0 problems of "how do I actually create an app and manage it?" Because of that, it was sort of a

gold rush to take advantage of this new platform and there were seemingly millions of permutations of stacks that you could run for your containerized app.

Luckily, over the last few years we've seen some standardization. At very least, Docker has unquestionably emerged the packed format. You've got Kubernetes, Mesos, and Swarm as the orchestrators of choice, Consul or [inaudible] for service discovery. So now that the building blocks are more set, the challenge is still around making all these different systems play nice together. Over and over again, you hear things as simple as taking a Docker composed file from the developer terminal and turning that into a running container for production, is a broken workflow. So there's no doubt that the open source community will continue to work to solve some of these problems.

I'll say, most of the juicy problems have been solved and most of the major platforms have been won. The one area I'm still excited about is solving multi-cloud. There needs to be a platform that gives companies the ability to deploy across clouds, across platforms and then manage and provide visibility into the health of those applications. Things like Kubernetes are an enabler of this and they'll serve as a distributed runtime, a distributed fabric to deploy applications across clouds. But then to actually gain visibility into the health of those systems is something that I think is really missing.

[0:12:12.6] JM: Yeah, you mentioned going from that place where you're putting out fires constantly to having more stable infrastructure and having this place where companies can experiment rapidly, the employees feel comfortable, they have a lot of leverage, things are decoupled, so they can roll out changes aggressively. I remember having a conversation like this. Last week, I interviewed Matt Klein who works on Envoy, which is a Lyft application proxy. He talked about how, when he arrived — I think when he arrived at Lyft, people were afraid to change code, because they were just afraid that this going to take down the entire system. They didn't feel like there was enough observability. They didn't feel like there was enough safety in a deployment.

Overtime, particularly, regarding Envoy, as he rolled that out, people got more comfortable, and it was a massive advantage. I guess I'll use that segue. If someone comes to you with an open source tool with traction — Let's take Enjoy, for example, because we just did a show on it. It will

probably air before this episode airs. When you look at something like Enjoy, it's this application proxy. It sits on all of your servers and infrastructure and it proxies requests so that you have a unified communication tool. It gives you a unified load balancing layer.

If somebody came to you with an open source that was like, "This has some traction, or this is deployed at a company." Is that traction enough of a reason to invest, or do you have to be able to see further down the line?

[0:13:38.9] LP: Look. I think this gets into a lot of the questions around open source in general. It's funny, this is timely, because we're looking at a similar project right now that's born inside of one of these web scale properties, and it's been battle tested. There are people using it. There are people excited about it. At the end of the day though, the goal with any technology, even though the world that we're living in is vastly different than it was maybe 10, 15 years ago, the same rules apply. The goal with any technology is to become a standard. Once you become a standard, you have these value chains that coalesce around these standards. Once you've codified your place in the stack, then there's a real unique opportunity to start extracting value. That's how really big meaningful infrastructure companies are born.

Green shoots of traction and adoption are a fantastic place to start, but the larger macro question is really about how applicable, how universal is the value prop and this is something that's universal. Not only is it universal, but it is a classic — Is it 10x better than the substitute? Because in an open source world, while anyone can take your code and improve on it, and so you want to be — Especially as an investor, you have to be very careful or tread lightly in a sense of not giving into this incrementalism. What might be good today, will it persist overtime and will the value remain of what you're doing?

I guess putting on the investor hat, whenever you're looking at a technology like Enjoy, what is its role today? What is the stack look like tomorrow? Is this something that becomes more valuable overtime? How could it basically ward off either homegrown solutions, different open source platforms? What is the stack really look like tomorrow? These are all questions that we're wrestling with as investors, and quite frankly, makes infrastructure investing a bit of a challenge today.

[0:15:37.6] JM: Absolutely. One of the ways you just framed this is how much surface area does this open source tool eventually take up? Obviously, the more surface area it takes up, you're building a market with a value of X and trying to capture some percentage of that market that's the adage for building platforms.

One analogy I think of here is Docker, and Docker is obviously become this fundamental technology to application development, and yet it seems like the business model is still up in the air. As far as I understand, the economics of the company, the burn rate, they still have plenty of time. There's no threat. I was at Dockercon and I remember having a lot of conversation with people like, "Is Docker going to get acquired by Microsoft?" It's certainly not obvious where they are going, and are they going to — Can Docker be obviated? Is the real platform Kubernetes?

It's interesting to look at that. Especially, I contrast that more narrow investments that you could make, where you're investing in maybe a very specific observability, like if some company came to you and said, "Okay. We're doing distributed tracing. We're going to master distributed tracing," or even like, "we're going to master Java distributed tracing," something very narrow and specific. In some sense, that seems like a safer bet, but maybe that's less of a venture investment.

I don't know. Where do you see that sort of like narrowness versus platform-ness playing into how — What makes a plausible venture investment?

[0:17:13.7] LP: Absolutely. This is the fundamental question I think, facing most infrastructure investors, is solve a very acute narrow problem. Basically, ingrain yourself in the stack and then expand. The question becomes, "When you position yourself in the stack, where are you more ripe to expend it to different adjacencies?"

Again, let's take the distributed tracing example. Distributed tracking could be potentially an incredibly strategic area, because you're instrumenting every part of the stack. While at RedPoint, we actually had a chance to invest and work with a distributed tracking company, they're still in Stealth. Once you're in the critical path, there's all sorts of data you're extracting, whether it's sort of in process logs, network calls that allow you to build a sort of comprehensive view of an application and its behavior and all of the underlying systems.

With that example, you eventually start subsuming multiple systems that you're using for observability, whether that's your traditional infrastructure monitoring, whether that's your APM tool, whether that's actual logging. That is a potentially incredibly strategic area, because it's all about where you're deployed and what data you have access to.

Again, in many ways, infrastructure investing is no different than a lot of other investing where data is critical, and data is the critical asset. What is your proprietary data source and what can you glean from that, and what value can you deliver to the end user? That's the critical question.

When you're looking down in the stack at more traditional utilities, same question. The reason past companies have struggled, I'd say, is because infrastructure vendors can subsidize the cost of the management deployment tooling and deliver a similar experience. Generally, it values even from the bottom-up.

This question is absolutely essential to what we look out when we're investing in companies. I hesitate to say we have an answer, but it's essential to a company's attractiveness when they have an easy path to extend either up and down or east-west in the stack.

[SPONSOR BREAK]

[0:19:18.5] JM: Are you ready to build a stunning new website? With Wix.com, you can easily create a professional online presence for you and your clients. It's easy. Choose from hundreds of beautiful designer-made templates. Use the drag and drop editor to customize anything and everything. Add your text, images, videos and more.

Wix makes it easy to get your stunning website looking exactly the way that you want. Plus, your site is mobile optimized so you'll look amazing on any device. Whatever you need a website for, Wix has you covered. The possibilities are endless, so showcase your talents. Start that dev blog detailing your latest projects. Grow your business and network with Wix apps that are designed to work seamlessly with your site, or simply explore and share new ideas. You decide.

[INTERVIEW CONTINUED]

[0:20:40.4] JM: Yeah, it makes sense. What about the RethinkDB example that happened recently? This was an open source business model that was a developer tool. It didn't work out. it had some traction in the developer community, but the business model didn't really pan out. I don't know if you read that post mortem, but what are the lessons that you take from the RethinkDB situation?

[0:21:05.8] LP: So when it comes to the question of open source and open source business models, I'm not quite a contrarian, but I am more optimistic than some VC's who just have given up on it and refused to touch it. My philosophy is that you can be a successful, thriving, open source company, it's just that the time horizons are elongated and that these businesses, through their growth curve, are much more fragile.

My hypothesis and thesis is that it's not that open source can't succeed, it's that it hasn't really been done right for several years. So to illustrate some of these challenges, I'm working on a new analogy, so bear with me a little bit. Tell me if I'm crazy. But the analogy is hunting versus farming. In the days of proprietary software, once you had a new technology and you found product market fit, it's really all about and going out and finding your customers as fast as you can and growing as quickly as possible.

As long as the market doesn't try up, so long as there's buffalo and as long as a new technology doesn't come to displace you, so long as a better hunter doesn't come on your territory, you're going to be fat and happy for a while. The point being, if you've got a great product and an eager market, you could extract value and grow really quickly and we saw a number of companies that did this really well through the course of the '90's and early 2000's.

With open source, however, it's a much more delicate song and dance. The problem is, of course, that your success isn't solely contingent on the merit of your capabilities as a product.

Or to finish the analogy, your capabilities as a farmer. So lets assume you're a great farmer, and everything goes well on your planting cycle. You use your seeds the right way, you use your fertilizer, the weather's perfect, even then it's going to be months before you can reap the rewards of what you sow.

But then there are also all of these elements out of your control; weather, birds — I really don't know what I'm talking about at this point. But the point of all this is, just like farming and raising a successful crop, building an open source company is really about fostering an ecosystem and nurturing your seeds. So in that process of actually building and open source company, there are so many things that are outside of your control that directly impacts your viability as a project or technology, or directly in business.

So people can take your code base and they can fork it, they can make it slightly better and they could capitalize your market. The cloud vendor can take your software and deliver it as a managed service and you're hosed. Again, to be with these points, it's not that these businesses can't succeed because there has clearly been examples of successful open source companies. It's that a lot of the time these business take much longer and they're much more fragile.

So in these businesses, I guess what I look for is, is there community and are the vendors building trust? The key with open source relates back to this idea of becoming a standard. So you have to believe that once adoption is universal and you've engrained yourself in the stack, or you support some unique workflow, then you an start thinking about monetizing. By the way, monetizing doesn't mean delivering something that's not out of the open source. But it really relates to delivering some truly differentiated developer ops experience that no one in the ecosystem can easily replicate.

So there are a few examples of this. I think HashiCorp is definitely one, Elastic is another. They are certainly few and far between, but again, open source as a business model has really become the status quo only in the last decade or half-decade. The proof of the viability of these businesses has really yet to be determined.

[0:24:47.6] JM: I can think of a couple of counter examples to the long lead time to profitability though, because I think about Cloudera, or Red Hat, basically, where you have the model of,

"We're basically going to have a consulting shop that is an incubator for an open source platform that is so complex to set up that you need consultants for it. They have this very just nice ecosystem that they can develop within a company where they have consultants, they have people working on the open source projects and there's a nice relationship there. I don't know. Does that still seem like a plausible — Maybe it's been a while since we've — I think we still see some companies in that mold, right?

[0:25:31.5] LP: In terms of the Cloudera model where you're sort of developing on the core, but also have an army of consultants and sales engineers.

[0:25:39.7] JM: Yes. Is that an appealing model as an investor? Is that too much? I think the consultant churn is like that makes it some execution risk.

[0:25:48.0] LP: I think it's a challenging model, and I think it speaks to the fact that it's — Cloudera was, what? Founded in 2008. It's been almost 10 years. They are without a doubt sort of the dominant force in open source analytics infrastructure. It's certainly a capital intensive model. It runs counter to the traditional software business, where it's high-gross margin, high contribution margin. There's an argument to be made where you can build a business with sort of technical domain expertise, where you are the trusted infrastructure vendor. You get to not only sort of reap the benefits of the ecosystem you own, which might be sort of HDFS, the Hadoop ecosystem, but also other things that come around.

If you've watched what happened with Spark, Cloudera has managed to co-opt apart to the value creation with Spark, just simply because, "Hey, this is open source. We already have account control. We're going to come in and we're going to setup a Spark cluster for you and also own all the training and support that comes around with it."

Again, this is another element that makes open source much more difficult, is when you have a company of Cloudera's scale that has this account control that's built trust with traditional enterprises. It becomes much more difficult to dislodge. Now, there is a question, "Can there be another Cloudera?" I guess if there's a technology that comes around that's as transformational as HDFS where the economies of learning are so steep where they can encroach, then, maybe.

[0:27:25.6] JM: Cloudera model sounds like you get kind of Accenture margins, where you have the army of consultants that if they were just consultants on their own, they wouldn't be able to make as much as they can at Cloudera, where they are Cloudera engineers, basically, consultants. Then, you also outlined how there's this opportunistic business opportunity where you can upsell people as the stack changes. It changes from Hadoop to Spark, "Oops! You want Spark? This is going to cost you a little bit."

When I think about the platforms, potentially, that could mirror that sort of evolution, I think of Kubernetes, I think of TensorFlow. These are complex — They're complex in a certain way. They're complex enough that I can imagine needing, or at least wanting an army of consultants. There are a lot of like Kubernetes, Kubernetes as a consultancy companies coming around, or we've got a fork of Kubernetes and it's got some proprietary layers on top of it. Do you find these business plausible? These are things like, I think, Rancher Labs, Apprenda is sort of doing this. Do you see these companies as analogs to the Cloudera type of companies?

[0:28:38.7] LP: Not particularly. I guess when you think about the companies you mentioned, there's sort of two schools of thought. We're going to come in and we're going to help you. We're going to support and service open source Kubernetes and we're going to take advantage of all the features that the open source community is adding on, and we're going to use the rate of innovation as a benefit and sort of draft off that and build a bunch of tooling and support and, basically, plug in the gaps, maybe in the management layer where the core Kubernetes community isn't keeping up.

Then, there's another school of thought where, "Hey, there's a bunch of enterprise features where core Kubernetes has missed the boat, and we're going to afford Kubernetes and we're going to have our own distribution." That, to me, is a challenging model in a cresting market where the rate of innovation is really fast.

In general, when you have a market and a product, or a platform that's developing as quickly as Kubernetes, I think standing in front of that title wave is really, really challenging. Instead of basically forking that product and putting your own spin on it, you'd be much better served building out the support ecosystem for that. Whether that's secret's management, or some sort of —

[0:29:52.5] JM: This is like Weave.

[0:29:53.7] LP: Yeah, exactly. I think those guys have a fundamental appreciation for open source done right and are basically building the tools and services to help you as a Kubernetes adopter be successful. I think that, to me, is the right strategy. Whether that's going to create a company of venture scale return, remains to be scene. I think that's challenging. I think it's probably, in terms of long term viability, that's probably a safer bet than working the main project and trying to live off that. Unless you're doing something that's so different and so value added, again, this sort of 10x nature, then I think it's challenging.

[0:30:38.3] JM: These companies like Weave, for example, who are focused on Kubernetes tooling, I think, basically, networking around Kubernetes and perhaps other container orchestration. Maybe, Mesos also, I think, and visibility. An advantage of those companies is they get to sort of ride alongside the Kubernetes ecosystem, but they're not exposed to the risk of building directly on top of the Kubernetes ecosystem, like breaking changes to whatever they have.

Also, they get the kind of opportunism that you mentioned earlier with Cloudera, where if they see a market opportunity, they have such intense domain expertise that they can perhaps seize that opportunity and then maybe their business still have venture returns.

I want to shift this conversation. I want to talk about the infrastructure layer, because this is the thing that is the most exciting to me right now mainly just because we have Amazon Web Services, we have Google Cloud Platform, Microsoft, IBM, Digital Ocean. This layer is exciting for a number of different reasons, but the one that I always think of is how it brings out the philosophical differences between Amazon and Google, which are the market leaders. We're seeing this — The market is evolving so interestingly, because Google and Amazon are kind of — They're different. They're offering the same — There's overlap, but they are offering very differentiated product philosophies. It sort of resists any particular analogy, but the closest analogy might be, "Oh! It's like Microsoft and Apple," or something.

[0:32:211.9] LP: Right. That's actually a very opt analogy. The best explanation I've heard about this, and to give someone — As weird as it might be, someone I've never bet, but someone who I respect a ton, Ben Thompson, a shout out on this, was right after Amazon reinvents. He had a great podcast around how do you differentiate between AWS and GCP, Google Cloud Platform.

The fundamental different was Amazon, in its essence, is a services company. Google, in its essence, is a product company. The way they expose their cloud offerings mirror that. What AWS does is they'll have primitives for a compute network storage databases, even higher order services around sort of real time streaming. They will always be exposed as sort of a standalone primitive that you can then go and build on top of. They'll be about 75%, 80% complete, and then it's on you to sort of integrate it in your stack and build on it.

Google, in its essence, is a product company. The way you interact with their services is through sort of a defined API that they control and they specify, and you consume those resources — Compute becomes an API resource, storage network. All these things are exposed through almost like a productized layer.

In that sense, that analogy holds pretty closely in a sense that one is a platform, one is a product. I think there's plenty of room in the market for both to exist. It's just a matter of what your developers, or what your ops team, I guess, in some instances, is used to and what they're trained on. Most of the market — Because Amazon created and starting, really earnest, in 2006, they were schooled on this stuff. So this notion of very basic boring primitives that you can build on top of holds. More and more, we're seeing companies gravitate to this sort of fully big productized version of what Google is offering. Especially with as much cloud as they hold in areas like machine learning, there's an opportunity for Google to meaningfully win customers over.

[0:34:13.8] JM: I agree with that. I agree with most of that. I do think Ben Thompson is — I love Ben Thompson. I love his podcast. I do think he gets wrapped up into the narratives of the companies a little more than he's apt I think. If you pigeon hole Amazon as a services — The thing is like these things are not mutually exclusive. Amazon's building out serverless. On top of their serverless expertise, they could easily build the type of, really, user friendly APIs that Google is building, like a Cloud Vision API — The Google Cloud Vision API, the Google trend.

[0:30:17.1] LP: Which they had.

[0:30:18.5] JM: Right. Okay. Amazon is building out those really simple APIs to identify an image, or something. I don't know. It's interesting, but the —

[0:35:04.5] LP: I guess the — Going back to these points. It's sort of what is your core competency, and what are you good at? If you want to go for sort of run of the mill boring infrastructure, build on top of that and use some of these higher order enterprise as services, Amazon is fantastic. If you want to basically — Again, this is speaking as someone who's never built an app on either of these platforms.

[0:35:31.4] JM: Me neither.

[0:35:32.5] LP: Take this with a grain of salt. Actually, this whole plug issue, we're taking it with a grain of salt as someone who hasn't written a line of code in about eight years. If you want to commit to sort of a Google way of doing things — If I was building a company, sure, that sounds awesome. Then, the Google approach might make sense. If you want to have, perhaps, a little more customizability and a little bit of more pain, then perhaps Amazon is the right — The vendor of choice in that sense.

[0:35:00] JM: Right. How do you see Microsoft? Where does Microsoft take market — As I understand, my simplistic understanding of where Microsoft takes market share is the enterprise companies that are moving from Windows to the cloud and it becomes — Microsoft already has a relationship with them. They're on Windows. It's a very natural transition. Is that the customer base that Microsoft is focused on, or is it wider?

[0:36:25.8] LP: If you spoke to me a few months ago, I probably would have said, "Yes, that's the customer base." To see the fundamental culture change within Microsoft around open source, around Linux, around cloud, it's been pretty phenomenal. I don't think we've seen a recasting of a company in a more dramatic sense than what's happened within Microsoft in the last couple of years. You don't want to forget that there are some pretty amazing engineers within those walls up their admin.

I would never handicap a race and not count Microsoft as a dominant player. With that said, I think what should scare Amazon and Google is the fact that from the up layer down, Microsoft is an incredibly compelling story. Most of the world is on Outlook and Excel, and these are tools of the trade that they use. When they could package and subsidize these apps with cloud platform, it's a really compelling story.

What wasn't lost on me was the fact that — I was interning for a hedge fund — When I was in business school and I was pitching — I don't know if we just say this online, but a short of VMWare, and it was stunning to me just how much of the world, specifically in between the coast, was Windows server and SQL server and how many of those shops were basically held captive by Microsoft.

When I think back to how much of the world is still running Windows, it makes me think that if Microsoft is successful in transitioning even a little bit of that world on to the Azure platform, they're going to be a force to be reckoned with.

[SPONSOR BREAK]

[0:38:03.6] JM: You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, glue code, lots of iteration, and lots of frustration.

The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today. That's exaptive.com/sedaily.

The Exaptive Studio provides a visual environment for using back end algorithmic and frontend component. Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3.

Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better. Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW CONTINUED]

[0:39:34.6] JM: Yeah, definitely. Talking again about that opportunism that you articulated so well around what Cloudera can do with Spark, it's so obvious that this cloud space is just — It's like an evolution from thing to thing every six months; Docker, and then Kubernetes, and then server — Not that people are massively adopting serverless, but the fundamental advancement that serverless offers, and we're just going to keep seeing these things time and time again. When you're a company like Microsoft, you are positioned for opportunism.

[040:07.2] LP: Yeah. Absolutely.

[0:40:09.3] JM: What about the other players, like IBM, Digital Ocean? Where do these fit into the — When you play out the way that the infrastructure layer advances over the next 10 years, do you have a feel for how IBM and Digital Ocean and perhaps the other smaller cloud players fit in?

[0:40:25.7] LP: So, whenever I'm looking at anyone that wants to duke it out with one of the major cloud vendors, first I say, "Good luck." But then I ask, "What's your edge? Or what can you deliver to a subset of customers that the big three can't?" AWS is obviously the 800 pound gorilla. It moves fastest. Google is Google. It has the benefit of probably the best engineers and distributed systems in IA in the world working on delivering these services and similar experiences to the modern enterprise.

Then there's Microsoft we chatted about and these guys know enterprise better than almost anyone else and they have a huge Windows and Office Install base to milk and transition over to Azure. So looking at any new potential rival cloud vendor, what can they deliver that's different or better? What kind of experiences will they give to developers that the big three really can't? So in my mind, Digital Ocean really nailed. They won through simplicity, transparent

pricing, and their expertise is really in marketing to developers. They run one of my favorite blogs and the content that those guys put out targeting their audience is fantastic. So they've developed an incredible brand for developers and that, I believe, can stand the test of time.

With IBM I'm not really sure. I guess they have Watson? So anyway, I guess that might be an interesting edge. The macro point here is that it's going to be the big three and they're really sucking up all of the oxygen out of the market and so in my mind, unless there's some massive discontinuity on the horizon that fundamentally remakes enterprise computing, I don't see how the dominance of those three big players can be challenged.

Now, to get back to your earlier point about serverless and what's next, to me serverless really signifies the conclusion of the first wave of software development as we've known it and it's giving rise to the second, which is all about IA or sort of this idea of real time distributed intelligence. So here's another theory, but back in the 60's and 70's, to write a program you needed to be an expert in processors, compilers, operating systems.

Then you had a couple of companies like Microsoft came around and abstracted a bunch of that complexity with operating systems. Then in the 90's you had the rise of Java and object-oriented programming and application servers and suddenly app development arrived in the enterprise. Then the 2000's, software development was further democratized through cloud computing, things like Ruby on Rails, Mongo, where literally anyone can, over the course of a month, go learn how to code and be a pretty facile developer.

So now, sitting in 2017, we're arriving at the serverless world where the logical atomic unit of compute becomes a function. So what that means really is, when you know how this unite of compute behaves, you know exactly what it's supposed to do, you can start stringing together these daisy chains of functions and build pretty sophisticated workflows and apps. So, in this world, the world we're coming to really is that any business analyst can write sophisticated apps that scale with requests because these distributed run times are actually built on top of AWI's and Google.

Concurrently, sort of what's happening, is that all the hard algorithmic work — a bunch of the core compilers and processors that were happening in the 60's and 70's, we're seeing a

renaissance and resurgence of that in the halls of universities and across bigger organizations like Facebook and Google. All this research and development really has to do with AI. To me this is sort of the second generation of software, which is all about building these intelligence systems. We hear a lot of people talking about AI as this fundamental new architecture, and what that means is that the infrastructure that supports these systems will be different.

Sure it'll borrow a lot on the innovation of this previous cycle, but also a lot of the building blocks will be fundamentally different. We're already seeing different processors, compilers, languages, frameworks, model serving systems. They're purpose built for these AI workloads. Again, the model for the future is Google, and if you want to see what the rest of the world will look like in five to 10 years, you look at Google.

So today, you look at Google's data center, they have a bunch of workloads that obviously run a traditional CPU's. They've got a bunch of MLAI oriented workloads that run on GPU's, but now they also have this thing called the TPU, the tensor flow processing unit, which is a special purpose built system on a chip for trading and serving machine learning. It won't be long until new systems and platforms emerge that support intelligence software of tomorrow. I think that's ultimately what I'm most excited about and looking out for.

[0:45:14.0] JM: Yeah, just to the show with Peter Levine that airs today from Andreessen Horowitz, and he was talking about this vision he has where computing gets distributed into self-driving cars, and the drones, and basically these are the data centers of the future. Everything just gets distributed among these. Then, the job of the cloud server gets reallocated to data intensive training, because you get the low latency responses from the data centers that are sitting in the self-driving car sitting outside of your window, rather than having to make a bunch of hops to an actual data center.

He was describing just how this movement of computers everywhere and into a peer to peer network system together with the pressure that machine learning puts on our current model of server architecture together will push us towards something that looks much different than what we have today. I think you alluded to that.

I do enjoy talking to VCs about infrastructure, because they have a different type of skin in the game when it comes to opinions about technology. If something new comes out, open up Hacker News, and you're an engineer, you can tinker around with it for a while. It doesn't really cost any money if you tinker around with it. You can build a little project with it. A VC has to put money where their mouth is if they think something is cool and they have to project how that thing will look against the environment as things evolve into the future.

There has been this massive explosion of podcasts by venture capitalists and some of them are not that great, but other ones are pretty entertaining, 'cause a lot of times they have to project themselves into the future and talk about computing in the future; it's fun, it's fanciful, it's futuristic, any other adjectives that begin with an F.

Explain your investment thesis when it comes to putting money into cloud native technologies.

[0:47:14.8] LP: Yeah. Look. I'm now at a fund called Amplified Partners, which is dedicated to working with technical founders that are solving some of the most complex challenges across enterprise infrastructure, security, other emerging enterprise technologies such as AI. At this stage, certainly, you have to have a directional sense of where the world is going and what it's going to look like. At the end of the day, you're just trying to find extraordinary entrepreneurs who are convicted enough to actually give their life to a problem that they're solving.

If I was a growth stage investor, I'd say, "Yeah, I have a directional sense of what the stack of 2020 is going to look like." Now, maybe taking a pass on the question, but I say I defer to the guys that are in the Ph.D. program at Stanford, or Berkley, and are working on problems that are 10 years out. Going and talking to them, thinking about the world, how they're seeing it. That's core to the investment thesis.

At the end of the day, I definitely agree with guys like Peter who say that computing tends to evolve and it tends to ebb and flow between centralized and distributed, centralized and distributed. The moment you start thinking you're at the end of an innovation cycle, another issue drops and things open up. I think the most important thing as a VC or an investor in general is just to keep an open mind to be people-oriented and to have them guide you.

[0:48:41.9] JM: Does the cloud native space feel over invested in right now? Are there too many companies that are chasing too few customers, but the venture capitalists are willing to go after them, because they are seeing things from different angles. They're seeing the expected value different. Perhaps — Go ahead.

[0:49:00.3] LP: I think, at the outset of every platform ship there's always an over investment period. Venture capital in general loves periods of rapid change because that's when new technologies emerge as standards and then you have entire new value chains organized around them, and so you get these winner take most companies emerge. So at the outset of any tech transformation, you naturally have a lot of smart people that are trying to fill gaps and position themselves to be these winner take all platforms.

I'd say, now, if you talk to investors, there's certainly an apprehension around cloud native. A lot of it has to do with that apprehension that's inherent to open source and just in general infrastructure. There are these two giant deflationary trends that have their paws all over IT, which is AWS and other cloud vendors, and open source. They're drastically driving the price down of all infrastructure technologies. You have that as a backdrop.

Around cloud native specifically, I'd say there was a platform company, Docker, and then there's a platform project called Kubernetes. There are companies that are trying to be sort of a Red Hat for Kubernetes, or sort of co-opt that project. I don't mean co-opt to the negative way. Inevitablly, there will be money to be made. Will there be a VMWare in this space? I don't know. Does Docker have a chance to be sort of GitHub or Atlassian type company? Absolutely. Then, there's going to be value to be made around deployment, around management, as there always is, around storage and networking.

Again, I think with every abstraction layer you move up, the opportunity to sort of extract value and build a larger value chain around yourself gets smaller, because it gets abstracted further down into the platform. With serverless, we're seeing a bunch of companies that are offering functional of lifecycle management. Again, I always ask these companies, "What do you do that the platform vendor doesn't? What workflow do you enable that doesn't exist today? How can you build a standard around that workflow?"

To me, these are the fundamental questions. What sort of new behavior do you enable that doesn't exist today? How do you shift the paradigm in your favor? Because if you're just tagging along, I'd say, sure, you could build something successful. Will it be venture scale? Probably — I'd say I have doubts.

[0:51:18.8] JM: The deflationary trends are interesting and that maybe you see less — For a given company, you see less capital going directly into something like infrastructure, because AWS — The deflationary trends of that. At the same time, if it opens up the budget, and I think you're seeing this. When it opens up the budget, companies are more liable to just purchase — If they see anything. If they see some APM tool that saves their developers a little bit of time, that time translates so easily to money, because the developer's time is so valuable that they just purchase it instantly.

That's why you look at that cloud native map that you made that I'll put in the show notes. You see all these tools that people do want to buy and they've got treat ARR properties. A lot of them have good amount of stickiness and they save developer's time and it's like, "Well, why will I not buy these?" Although, at the same time, I think you make a really good point. At that point, are they venture investable? Maybe not. Then, that supports the further trend of the breaking down of the wall between seed investing and venture investing, because if something is not going to deliver venture returns, then you need to give it a different style of investment.

I think it's the same thing that's happening on consumer companies, where you have smaller investments with perhaps smaller expectations, and maybe perhaps a frothier environment, because it's harder to tell what's going to pan out and into what degree it's going to pan out, to what degree it's going to be profitable. Yeah, for the past three years where people have been talking about this, whether you talk about, "Oh, it's a bubble, or it's a risk bubble," or whatever. I've always just — Throughout this entire time, I'm like, "No. It's just a change in the economics offered by venture returns on technology investments."

[0:53:04.0] LP: Just to go back to a point you said earlier around developer tools and how this sort of tried and true things, you can't make money selling to developers. Yes, it is difficult to make money selling to anyone who doesn't own the budget. In a world where software is not

only core to your business, is your business, tools that sell to developers are no different than ERP. These are tools that help you run and build your business.

[0:53:27.8] JM: That's right. Yeah.

[0:53:29.9] LP: Especially talking about a world where, in my thesis, if serverless opens up programming to not only your front-end Java Script dive, but also guys in marketing, and guys in sales, code is going to be everywhere. You're going to need tools that support, and monitor, and make that code better or optimize it. Developers tools will seize to meet developer productivity and they'll spend the entire organization.

Again, I think part of it is a taxonomy problem around what is a developer tool actually mean. I think the definition need to be refined as software eats more and more parts of the organization. I definitely have some thoughts about this, and all of Silicon Valley is obsessed with this notion of the power law, where you've got one company that pays for all these failures, and it's this failure culture. Specifically an enterprise, and when you're looking at companies that are building strategic IP and when you're dealing with companies that have a bounded total addressable market for what they're doing, the outcome of distributions, in my mind, and some of the data backs this is up, is actually a much more into a normal curve as supposed to a power law distribution.

If you look at companies that have been ventured backed and produced outcomes over $2 billion in the last decade, there's something like maybe 10 of them, in the teens of companies between $1 and $2 billion. An alliance share of the returns are really made in the companies between sort of $200 to a billion. By the way, the incidence of zeroes is far fewer as well.

When you look at that distribution of outcomes, there are a lot of money to be made working with companies that solve a particular problem for a particular customer segment and are super successful at servicing and delivering value to those customers as supposed to something where everyone is feeling this pain and you're servicing everyone. That's core to my belief as an investor.

[0:55:28.7] JM: There are all these companies making the “digital” transformation, whether you’re talking about a company with a ton of researches, like GE, or maybe State Farm, or Legacy, companies, like, I think, insurance companies, fertilizer companies, these companies where they’re finding out that they’re ultimately software companies. Maybe they’re software plus — They’re a software company that sells insurance, or a software company that sells fertilizer, or whatever.

What areas of the cloud native stack — Are these types of companies adopting? Are they adopting it aggressively? Are they destined to adopt cloud technology, or are they just going to stay on some on-premise thing in perpetuity? What’s your sense for how the large enterprise is shifting? Do these companies just die eventually and then they get supplanted by companies that think about themselves as software first?

[0:56:24.5] LP: I don’t think so, because at the end of the day, what’s missing, or what’s — Not missing, but what’s critical in building any businesses, deep domain expertise of your customers, your vendors, everyone in the value chain. Software, at the end of the day, is an enabler. It’s a distribution model. It’s how you deliver your value. It’s how you instantiate value, but it’s not — The relationships you have in business, I think, supersedes software, or the delivery of software.

I think, in a vacuum, if you have a company that’s — One; that’s software oriented, that understands the value of going cloud native. That means they could deliver software faster. That means their application is more resilient. That’s what it all translates to at the end of the day. Cloud native for cloud native’s sake is a waste of time. Cloud native for having an application that’s more resilient so you’re not Delta or United that grounds multimillion dollars-worth of flight, because you have a computer errors, and cloud native, because you deliver faster and innovate faster than your competitors. That matters.

There’s a business value associated with all of these, that’s why it’s important. It’s not important because you’re an ops guy and you’re excited about Kubernetes and like, “Hey, look. We’re running Kubernetes, but none of this shit works.” It matters because there’s actual business value associated with all of these technologies. What we look for zooming outside of the cloud native ecosystem — What we look for when we talk to entrepreneurs who are operating

application companies, or building a SaaS company, is do they understand the ramifications of CICD? Do they value being able to deliver more and faster than their competitors? Do they understand the value of actually being first to market? That's how all these things translate into actual business value.

That's what's critical here. It's not the fact that you're using the latest and greatest technology, it's the business case for all these stuff that matters. That's why it's important for, not only, VCs, but everyone to understand what's out there and understand the infrastructure stack that you're building on top of — Understanding how it evolves, and understanding what you want to accomplish.

Again, the most important takeaway is — I had the fortune of working with the guys at Datadog very early back in 2011, and when I talk to them now, I ask them, "Why have you guys succeeded while a bunch of monitoring companies have since either struggled or failed?" They say, "We haven't written a lot of code that was anything more than our customers actually derive value from."

At the end of the day, it's all about building value, understanding what value you're bringing to the table and capitalizing on that. To me, that's why all these stuff is important, it's not important in and of itself, it's a means doing that.

[0:59:05.6] JM: Okay. I think that's a great way to conclude. Lenny, thanks for coming on Software Engineering Daily, it's been a pleasure talking to you.

[0:59:09.4] LP: Thanks so much, Jeff. It was a lot of fun.

[0:59:11.0] JM: Okay. Great. Yeah, I really enjoyed this.

[END OF INTERVIEW]

[0:59:20.4] JM: A few things before we go. If you like the music on Software Engineering Daily, you might like most recent album called Commodity, which features a lot of the songs that air on this podcast. My artist's name is The Prion on Spotify, Apple Music, and Amazon.

Also, Software Engineering Daily is having our second meet up, March 9th at Galvanize. You can find details on our meet up page. Finally, we are about to post our second listener survey, which is available on softwareengineeringdaily.com. This is your opportunity to have your voice heard and let us know how we can improve.

This data is super valuable to us and we'd look at every single response, so please take the listener survey at softwareengineeringdaily.com. Thanks for listening to all these announcements. We'll see you next time on Software Engineering Daily.

[END]