EPISODE 1486

[INTRODUCTION]

[00:00:00] SPEAKER: Venture capital investment has continued to flow into technology startups. No one builds technology from scratch. There are cloud services, software libraries, third-party services and software platforms that modern entrepreneurs must adopt to build their products efficiently and quickly. These layers of infrastructure are a key area for many investors. Tim is a partner at Menlo Ventures where he invests in cloud and data infrastructure, AI/ML, and cybersecurity. He joins the show to discuss what goes into a smart investment.

[INTERVIEW]

[00:00:35] JM: Tim, welcome to the show.

[00:00:38] TT: Thanks for having me.

[00:00:38] JM: You worked at Splunk as CTO, and now you are doing investing full time. And Splunk is a company that, you know, if I would have told you, “Hey, I'm building a logging company or monitoring company, observability company”, even back whenever Splunk was founded – how may years ago, Splunk was founded? 10 years ago, something like that.

[00:01:06] TT: No, I think it's more than that. I think it’s 2006, somewhere in there.

[00:01:09] JM: So, even in 2006, there's a lot of monitoring. I think it's an example of the kind of company that it's not always easy to parse, what companies are going to be a breakout success, especially when they resemble existing products. Another example is, Datadog, it's kind of a mystery why Datadog was the breakout success, or you ascribe many kinds of theories to it. But I think this is one of the interesting questions in infrastructure investing is products that look redundant, or they look like their existing products, or they resemble existing products, they just may have some categorical differentiation there. So, I just use that as a preface to when you are assessing an investment, and it resembles stuff that's already in the market. Another example would be like the no SQL category. How

do you identify the successful investments that look like kind of the same company that has already been in existence?

[00:02:19] TT: Yeah, I think you're looking for a couple of things and I'm happy to touch on why did Splunk become Splunk in a subsequent question, if you want. But you're looking for a wedge, right? And when you're looking for a wedge, there's different ways to create that wedge. There are pricing wedges, there are feature wedges, there are capability wedges. As coming from an engineering background, what I'm looking for is folks that are building better products, that's probably an order of magnitude better than an existing solution. I don't think too much about price at that point. Because if something's an order of magnitude better, people will pay for it. But it has to be that large. It has to be a humongous step function for it. It can't be something that's incremental, because the cost of switching is pretty high, especially in the IT infrastructure, right? Because things like say, pick on Datadog or Splunk again.

Once you put those in, they tend to stay pretty resonant, right? Because they're going to be in the critical path of your data architecture, or how you run the business. So, that's where the really crucial. The other thing I look at is just founder market fit, right? Maybe that company and that founder don't necessarily have the immediate solution today to say disrupt Datadog, but they may have this vision that they can clearly articulate over the next five to 10 years, that's going to get you there. I'm willing to go on that journey with folks, and that's really what you're signing up with venture is, you want to go on the seven to 10-year journey with folks to have some magnificent outcome in the future.

[00:03:46] JM: A company that you're on the board of is Julia, and Julia came out, it's a language, it's very popular. But it's quite hard to monetize a programming language. So, that's another area of infrastructure investing that I'm curious about. Tell me when you look at a language, how often is a language monetizable? Was there something unique about Julia that made it an investable system?

[00:04:20] TT: Well, I mean, obviously, there's a lot of literature out there around how it's faster than Python and people talk about 10 to 100 times faster for different workloads. But as you pointed out, and I totally agree with you, investing in language can be pretty ugly. I think there's not a lot of like economic outcome from necessarily investing in, say, Python 20 years ago, right? What you're really investing in is the ecosystem that gets built around it.

So, what I find really compelling about the Julia guys was, they're obviously working on the language, but that's just part of the equation. The rest of the equation is what they have called Julia Hub. And Julia Hub is a number of things and now they have VS code integration there that lets you build code and actually deploy it on their cloud services directly, which is tuned for Julia style workloads and they have GPUs and obviously humongous multi core setups.

But then there's a whole suite of applications that sits on top of that setup. And so, there's something called Julia Sim, which is a simulation suite that is very, very popular. They're getting into other markets like the EDA market and there's something called Pumas, which they work with very closely with. It's actually a company that runs on Julia Hub. It's really an ecosystem, not a language. So, what I'm investing there, is more of the platform. In the Julia platform, the applications that sit on top of it, rather than the language. It just happens to be that the language is at the core of it and enables those applications and it turns out, that's not very different from a lot of other platform companies say, even like Splunk. Splunk started off as an IT log querying tool. And then suddenly, it turned into the Gartner MQ leader and security sim. The security sim product was just a series of applications that sits on the platform. So, I'm looking for these types of platform ecosystems primarily.

[00:06:10] JM: If you take Julia as an example, what is required to take an open source technology and bring it to market to the degree where it's actually large enough to reach IPO. I think about Hadoop or Spark, with – now, they have lots of tooling built around Spark with Databricks, and they manage to have all this infrastructure that is compelling enough to grow to massive scale. What's the system to do or to drop a playbook for taking an open source project and getting it to be large enough to develop into something that is IPO-able? What would that playbook look like?

[00:07:05] TT: Great question. So, I think step zero is make the community as large as possible, as quickly as you can on GitHub. I look at a lot of companies that have open source in their roots, and to be honest, one of the first things I do is I plot the number of contributors and the number of folks that have started that on a series of line graphs, versus other really popular open source projects that have gone on to monetize well. So, you can imagine looking at how Kafka grew, looking at how Spark grew over time, and trying to see if they're on the same trajectory. It turns out, I actually did that with Julia. I have a medium post on that, and people can find on Google pretty easily, where I plotted the growth of Julia in terms of contributors and activity over time and it sort of parallels Spark almost precisely, interestingly enough.

Like I said, step zero, grow the community as large you can, get a critical mass, so that there's a point where you have a really robust, active community. If you have that community, that probably means a lot of deployments out there. Those deployments are probably starting to pop up in a lot of really critical situations across enterprises, startups, medium-sized companies, and even government now, right? You probably couldn't have really said that, honestly, five years ago, seeing a lot of open source in government, and they're really starting to open up there. So, once you have that community built and people are using it, it's not shocking that there's a lot of folks who want to start to pay you to help them use that software, right?

So, there's a lot of ways you can monetize. You start to sell support, professional services, where you bring people in, and they sort of sit with your deployment and help build it out. But there's another path, which is really interesting. And frankly, I'm not so sure exactly how I feel about it yet but it's proven to work well, which is the sort of private version of your software starts to look very different from the open source version, and you can obviously say that for Databricks. Spark, and open source on GitHub looks almost nothing like what you would get from a commercialized version in Databricks. So, there's obviously that path to it as well. The open source sort of fanatic in me, I'm not sure how I feel about that necessarily, because I cared deeply about open source in the community. It's pretty clear Databricks has done quite well with that model.

[00:09:18] JM: Let's talk a little bit more about your experience with Splunk. So as CTO, were there a lot of situations where you had to – or maybe you could point to a situation, because there are two distinct kinds of situations I'd like to ask you about. One is a reactive situation where you have to put out a fire and maybe that could be a management fire or an actual technical fire. The other is a proactive situation where you have to draw up a roadmap for a future feature or a vision. Can you walk me through what those look like as a CTO from the CTO's perspective?

[00:10:04] TT: I think my role as CTO is a lot different than people who – or my role was, who traditionally had that title. I'd say there's a couple types of CTOs. One is more the field facing CTO who's kind of like the evangelist that's out there, talking about the vision of the company and sort of rallying developers and drawing up interest. You're probably speaking at a lot of vendor fairs and developer conferences. That very much was not me. My role at Splunk was probably more what you would call a CPO at most companies who've run product and engineering at the same time, which is

what I did. I just don't like that title, to be honest, because I'm more of a technologist and an engineer than anything else.

So, it's just very important for me to maintain the CTO title. I was very much still in charge of product and engineering. I also had IT and security and design in my team as well. So, I spent more of my time thinking about the product roadmap, and trying to think about the five year, the 10-year plan, and obviously, that was baked in a lot of technology thought. I would say that was probably 70% of my time and the rest was sort of a combination of selling software, and putting out fires. I mean, it's hard to avoid sort of putting out fires either from a product operations standpoint, when you're a cloud company, or a people capital standpoint, when you have 2,500 people on your team, which is where I was at. You're going to have these issues. And so, you're balancing your time and doing the 70/30 thing, in hindsight, may not have been the right balance. I think I should have dialed down the working with the technology teams or drawing roadmap or prototyping things. But that's just who I was.

Looking back on it, the company grew sizably while I was there, so maybe it was right. But I would have liked to have spent more time with customers. I would have liked to have spent more time just understanding their needs more, rather than sitting in the office trying to speculate what their needs were. If anyone's out there listening, don't forget your customers. They're number one, they're everything to you. So, spend more time listening to them than to yourself.

[00:12:08] JM: So, you mentioned you could discuss this a little bit more, what was it that made Splunk, Splunk?

[00:12:15] TT: It's really interesting and I'm glad you asked that. There are a lot of tools out there that could process log data and you said that, as well. But what Splunk did was uniquely was a couple things. One is they were targeting the sort of sysadmin IT, admin user, who was sifting through data, trying to find why posts went down or applications died. And what's really smart, there's, if you think about who the persona of that person is, they're probably Unix or Linux junkies, and they're going to want to express the query in a way that makes most sense to them. And that's not necessarily SQL, which is how a lot of people thought about that problem.

The thing that I think was killer about Splunk, at the time was the query system is essentially Unix, Linux command line stuff where you're piping commands together. That's really Splunk' language called

SPL underneath the hood. That's pretty magical when you're targeting IT user, right? IT users want to be in the command line, they want to be in the terminal. If you can suddenly express your query as what looks like a Unix command, just tons of pipes, and redirections, together with some functions as commands sprinkled in, that's pretty unique. Because as the admin, you're there in the terminal firefighting, and now you can also query your data at the same time without leaving the terminal.

The way that it actually processed the queries was very powerful as well. I mean, behind the scenes, it's doing a MapReduce, similar to sort of how Hadoop does MapReduce as well. It's doing quite efficiently. It's done in a really elegant, distributed federated way. And so those two things together, really made it take off and they did a fantastic job marketing. They understood who the user and the persona of the user was, and they attack them directly.

So, the way they would communicate to the user was ultimately the buyer was through their language, right? You would see all the Splunk t-shirts running around that were very geeky. And sometimes the t-shirts would have Unix command line sort of inside jokes in them, or they would really just speak to the hearts and minds of who the user was, and they just built this really rabid following, based on that combination of fantastic product and fantastic marketing. It just sort of took off from there and the users took Splunk down into the paths of security and IT ops and then observability. It wasn't Splunk necessarily that did it. But going back to my earlier comment, if you listen to your users and what they want, and pick your software in that direction for them, they'll follow if they really care about your product, and I think Splunk nailed that.

[00:14:50] JM: Has there been, when you look at the history of Splunk, were there any technological shifts, or pivot Little changes in the infrastructure landscape that you think the company could have handled better?

[00:15:06] TT: Yeah. Probably a couple of things. First is obviously, the cloud transition, and the movement to SaaS software in the Cloud. That's really why I was hired, and I mean, there's a couple of attempts at it, prior to my joining Splunk. But there's a way to solve that and there's perhaps a way to not solve that. And the way to solve it sort of elegantly is you want to be cost efficient. But you also want to introduce sort of techniques for modern databases in the cloud. To do that, you have to do a bit of open-heart surgery on the patient at the table. So, for Splunk, which was historically, this large, monolithic binary, written in C++, to put it in the cloud, you have to sort of break it down. It can't just be

a monolith, that's single tenants, right? Your capex costs are going to go through the roof, if you do that. You have to break it down into a massive multi-tenant system comprised of services.

So, you have to be willing to go on that journey. And it's going to be painful and it's going to be three to four years and it's not as easy as it sounds, because you can quickly innovators dilemma yourself into not doing it, and not be willing to take the time to go do that work over a period of four years and be willing to perhaps, take the hit on features on the mainline product and able to do that, because it's impossible to do unless you double the size of the teams. And even then, that might not be enough.

I think waiting to bite the bullet and do that, I think it took a little bit longer than it should have. That work should have probably commenced well before I got there, in the way that I described, which like I said, is a long journey. I think the other one was just waiting to get into observability. You mentioned Datadog earlier. I mean, it's pretty clear that observability was starting to take off, and it was going to be a thing. I think it took a little bit of time to get into it. Ultimately, we did and we acquired a number of companies, but by then Datadog had established itself pretty well. I still continue to believe there's a stark difference between the products and different buyers do continue to buy it. depending on sort of the size of the company. But Datadog, it’s pretty clear as a runaway success and awesome job by them and I think they're an amazing prototype for how do you build a cloud SaaS business these days.

I mean, it's completely PLG, it's developer focused, you can swipe a credit card, they have the free trial thing just nailed. I think they really figured out this idea that for that buyer, you have to have elegant design. You can actually hide a lot behind elegant design when your buyer is swiping your credit card. Because what you're trying to do is accelerate the process of buying that software and make the onboarding as quick as possible. Because I think people who are integrating things like Splunk or Datadog, they're developers, obviously. I say this as a proud developer. Developers are lazy. I would be surprised if you didn't agree with me.

So, when you're trying to add login to your application, that's not probably the first thing you're trying to do when you're coding it up, right? It probably is towards the back half of what you're doing, you're starting to do that when you're thinking about shipping. So, what's interesting is, I think, for a lot of next generation cloud services, you want to be as quickly integrable as possible, because the developers are lazy. So, if your solution integrates quickly, and the API's are super clean, and the UI is super smooth and easy to understand, if you can have time to value be five minutes or less, you're winning.

So, I think data dog just absolutely nailed that. Absolutely. And you can see it in their results and look at the market cap these days. So, if you're starting a company these days, honestly, I point to Datadog. Look at what they've done, look what they've done with PLG. Look what they've done with design, usability, UX, it's fantastic.

[00:18:51] JM: There is occasionally a moment in the world of infrastructure where something entirely new comes up, and it's difficult to figure out how to orient a investing thesis around it. And I think probably previous examples included data infrastructure, data engineering, but crypto is obviously a completely new animal, particularly crypto infrastructure. I imagine that as an infrastructure VC, there's two challenges there. One, you have to admit that despite crypto feeling like infrastructure, it's very, very hard to understand from even as a position of somebody coming from a distributed systems background, cloud infrastructure background where you have a lot of understanding of distributed transactions without actually getting into being crypto developer and going to crypto conferences and putting yourself waste high into crypto information. It's pretty much impossible to be able to understand what's going to make a good crypto investment. I'd love to know how you're navigating that shift, particularly as it's clear that, at a certain point, the crypto infrastructure world, and the conventional infrastructure world is going to intersect. I imagine you want to be ready for the intersection.

[00:20:22] TT: So, we're spending a lot of time here at Menlo thinking about Web3 and I do think that can be a full time, or it is a full-time job already, just because the landscape is evolving so quickly. What we're doing is there's myself and a few other folks who are spending a lot of time on a daily basis, meeting as wide a sort of spectrum of folks from the crypto Web3 scene as we possibly can. So, we're meeting crypto VCs, DAO, creators, folks doing layer zero, layer one stuff, all the way up to trying to build the next NFT marketplace. So, what we're trying to do is build a very sort of focused investment thesis to figure out what the angle of attack is going to be on crypto before we start doing a ton of investments. And disclaimer, we haven't done any large-scale Web3 investments yet, but we have a pretty well-defined hypothesis around where we want to enter.

And I think you hit the nail on the head, I'm sure you'd be pretty shocked if I didn't tell you that we were going to go after a very picks and shovels approach, right? I'm not really as interested in say, an NFT marketplace that has to compete with OpenSea, as I am in the guy is trying to do the low-level platform work at the infrastructure layer. So, I think there's just this massive opportunity where tons of crypto

companies and what the companies are going to be built on that infrastructure, and there's just a massive economic outcome that's going to come from that.

But to your point earlier, you could spend a lot of time thinking about it. But surprisingly, the more time you spend talking these folks, there's a lot of parallels and sort of, you can see the similarities between the existing infrastructure companies and what's trying to be done in crypto and Web3, right? There are companies popping up that need to do data processing against blockchain data. Well, how do you get access to that data? Well, now there's other companies that are becoming data management platforms, on the sort of infinite number of chains are going to pop up. And those companies are going to do well, right? Because standing up and getting access to chain data is not that easy, right? So, there's going to be infrastructure companies who provide that as a service.

Well, turns out now there's ETL that has to be done on that data. There's warehousing that has to be done. There's reverse ETL that has to be done on it. So, it's almost like a parallel dimension of enterprise software that'll be created on crypto and that's why I think you're seeing this increase in opportunities. It's almost like now we're multi-dimensional, in enterprise software in the same way that the multiverse has been done and all the Marvel shows. I oftentimes think about that, right? There's like, parallel storylines for Spider Man and Doctor Strange that's sort of happening with ETL systems and data warehouses in cloud and Web3.

[00:23:04] JM: Yeah, I've been thinking about that a lot. It's such a predictable x for y, what is the cloud data warehouse for blockchain.

[00:23:13] TT: It's pretty easy to reason about. It's actually not going to be – I mean, your needs don't change. You're just looking at a different kind of data. So, in the same way that Splunk, or any other logging system or even OLAP system indexes data, you still have to index blockchain data. You're going to want to have indexes on every column to query the data and slice and dice it in multi-dimensional ways. It just happens to be that it's chain data. And yeah, it does add some complexity. But at the end of the day, I think the problems aren't massively different. You're building it for a slightly different buyer with different data types. But a lot of the product fundamentals, the sort of computer science principles that you apply when you're developing the software, don't really change that much.

I am actually more fascinated by the idea that the user is changing, not the problem. The people who are going to build on that picks and shovels software are very different from the folks that are building on snowflake. If you sort of start to accumulate that even further, who are your biggest snowflake users? Well, they're probably sitting in Looker or Tableau, and they're slicing product analytics data or e-commerce data. It's very OLAP and multi-dimensional.

In the crypto world, it might be a company like I'll pick on one that I really liked, Chainalysis, right? They're doing sort of forensics on blockchain transactions. Well, how do you get access to that data? How do you mine it and query it in a way that's efficient? Well, Chainalysis probably has their own team doing it, but there's going to be other companies who need to do similar types of analysis. The user that does that is probably not the same guy that's sitting in Tableau or Looker, it's a different demographic, they're probably solving different problems. They come from different parts of the worlds, just massively different backgrounds. But they have similar sort of queries, it's just different datasets. I find that fascinating, to be honest.

[00:25:07] JM: What was the OLAP store that Splunk used?

[00:25:10] TT: We never tried to do be OLAP, to be honest. I always wanted us to be OLAP. But that buyers different. The buyer or the OLAP is more of the business analyst and product managers and folks of that nature. The Splunk buyer persona, in the later stages, obviously, is the CISO. It's VP of Engineering because they need to deploy observability. It's the CIO and the IT Ops, AI Ops sectors. So, those guys aren't necessarily and gals are not the OLAP buyer, right? It's the business analyst that's the OLAP buyer. So, it's very attractive to do it, I wanted to do it. The problem is, is you have to consider a lot of variables. Is the sales team equipped to sell to that buyer? In a lot of ways, we weren't. We didn't have those relationships. We're very deeply ingrained in the CIO, CISO world, and that was great for us. And it would have taken a lot to sort of stand that up. But I can definitely say, there's no reason why the Splunk product couldn't do it in the future. I think, it's positioned well to do it if it came down to it.

[00:26:17] JM: I just meant there's some kind of backing storage system for like high volume metrics and observability data, right? I guess the better question is like, what is the data warehouse that backs?

[00:26:30] TT: So, the real capability of OLAP probably didn't show up until we've started to buy observability companies. So, we acquired a couple of really large companies, SignalFx, and Omnission, and number of others. SignalFx provided a really robust metrics engine. The guys inside of Omnission, I think they're not shy about using open source OLAP technologies. Those are serving a really good sort of purpose in the observability space now. But that happened later, right?

Going back to the earlier days of Splunk. I mean, at the end of the day, what is Splunk? It's a time series index, right? So, you really don't want to use a time series index to query things into sort of cubing OLAP way. What you want is indexes all over the place, you can slice and dice the data and reduce the sort of surface area of that data corpus you're scanning. Unfortunately, that's not what timeseries does. Time series is about sequential scanning, and a lot of your queries are predicated on time, rather than dimensions in the data like OLAP.

[00:27:27] JM: Have you seen any significant shifts in how infrastructure companies are built after the standardization on Kubernetes?

[00:27:39] TT: That's a great question. Obviously, everyone's building in containers. They're using STO, Envoy or, any sort of derivative thereof. These things tend to be pretty transient. So, I may not, there's Linkerd, and all these things and there may be something newer since the last time I thought about this problem, before I came a VC. But I think, I'll have a maybe a different, slightly different response to you is, the more companies I talk to you, the more frustration I sense with the developer community with Kubernetes. It solves this massive orchestration problem, which is very real, and there needs to be a really fantastic solution. So, there's no doubt about that and containerization has done wonderful, thanks for software engineering. It just made it so elegant and easy to ship software.

But the amount of overhead in terms of how you think about deploying an application on Kubernetes is pretty significant. Now, you're thinking about external ports and internal IP addresses and how to map them and Daemon sets and pause. It's just, it's pretty hairy. So, it's interesting, like Kubernetes felt like it was curing cancer four years ago, or five years ago. Now, folks want more velocity in their shipping, DevOps has become pretty heavy handed, and folks want to find faster, cleaner, more elegant ways to solve the same problem. So, that's more of what I hear from folks, when I talk about Kubernetes. How to make it easier and reduce the amount of friction. There probably be in some of you probably solving that problem right now. How do you build a cleaner, simpler, smaller Kubernetes? I'm sure it'll happen.

[00:29:16] JM: There was a layer two cloud company that I spoke with a while ago, that is now very successful. And when they were shopping around for their investment capital, they had some trouble convincing investors who didn't think that layer two clouds were a viable opportunity. Now, we've seen there's probably at least two or three-layer two cloud providers, post Heroku, that had been billion-dollar valuations. I'm not talking about Snowflake. I'm talking about just Infra, general Infra. But I guess I'd like your perspective, is layer two cloud still a viable opportunity? Has the opportunity kind of been sucked out of the room at this point?

[00:30:09] TT: I don't think so. I think there's still opportunity, particularly because there's a pretty robust interest in being multicloud, as much as possible. So, layer two in the context of Heroku. I mean, there's a lot of companies trying to do that. In a lot of ways, those are popping up as what are called DevOps in a box companies. Definitely see a lot of these companies. It's really hard to differentiate one sort of DevOps in a box companies versus another because they're all solving fairly similar problems. I mean, there's a lot of value in those problems. But being able to differentiate between them is, I say, fairly difficult.

What I think is more interesting, and I haven't seen anyone really solve this well yet, is, how do you build a truly portable application across the major cloud service providers, and then be able to carve yourself completely portable, and completely cloud agnostic? It's hard. Because no matter how hard you try, if you start off building your application on, say, AWS, let's pick on them, there's just going to be places where native services within that cloud service provider pop up, without doing it on purpose. It's just going to be there. So, once you finally get to the point where you think you're actually portable, you're not.

I saw this problem pop up a lot in my past life. We would have Splunk running in leveraging some cloud service provider solution. But then you would run into some theater or sector where they're just adamant that your solution exists just in that cloud, where it's local to data that they're collecting, or they want to ship into Splunk and they just have to be in that provider. So, for me to lift and shift the solution from one cloud server provider to another, is tremendously high in terms of cost and man power to make that happen. If there was a sort of layer two service that could provide that, I'd be really interested in it. Because this problem exists all over the place. I mean, if somebody tells me that their solution is 100% lift and shift and portable, I would ask questions. I hear it a lot. But then you sort of peel back the

onion a bit, and it's not quite there. So, between that completely portable cloud agnostic sort of framework at the layer two level, and then this DevOps in a box set of offerings, I think there's tons of opportunity.

[00:32:34] JM: Why do you think Google Cloud has been unable to catch up to AWS?

[00:32:44] TT: Well, AWS has a number of advantages going for it. One is the breadth of the offerings. It's just imagine your developer for a second, right? Again, going back to what I was saying earlier, you're doing a lot of services together, you're adding some logic, you're doing DevOps, every service, or nearly every service, you can sort of need is there, right? I mean, there's, I don't know, 250 offerings, I want to see. Someone can keep me honest, I don't know the exact number. But it's significant. So, you can imagine, anything you can think of, if it's there, it's close by, you're going to use it. So, it's so hard to leave that ecosystem.

I think the other thing that's interesting is just the brand awareness. I mean, when you think about building a cloud product, generally, people tend to think about Amazon sort of first. I mean, obviously, Azure and GCP are there. But we think about cloud utilities, Amazon is sort of top of mind. There's a lot of inertia and momentum behind that and it's hard to break down. I think the other thing is, I mean, there's this sort of idea that GCP is hard to use. I hear that from folks here and there. I'm not representing that as my opinion. I'm just sort of regurgitating what I hear from talking to folks.

So, I think, unless you can sort of solve that problem, as we're talking about with Datadog where you make things easier for developers to use and make it integrable quickly, you're going to have a hard time catching mindshare and dollars from developers. If you're GCP, I think I would caveat that with also, what I hear from folks is they feel like the GCP services tend to be faster and more scalable, right? So, if you have this superior technical solution, you would imagine, people sort of gravitate towards that more as perhaps you're insinuating. But if it's hard to use, then you're not going to get people. Like I said, developers are lazy. That goes back to another theme that I talked about a lot with folks is oftentimes engineers who start companies, they tend to want to build the fastest, cleanest, smallest solution out there, and they think that the best technology wins. Certainly, in enterprise, I don't think that's true at all. I think the technical solution does matter and needs to work well and needs to scale and so on so forth. But doesn't necessarily always have to be the best. The go to market team is incredibly important. The marketing around the product is incredibly important. The customer success

part is incredibly important. And then there's the product. I think a lot of folks sort of missed that, and to be honest, I made that mistake for a number of years earlier in my career.

[00:35:16] JM: For 14 years, you were a VP engineering at Yahoo. I'd just be curious, it kind of feels like the industry has internalized a lot of learnings from Yahoo. I don't think there – I mean, Yahoo was insanely successful, of course, but kind of the way in which it got backed into a corner, and was kind of unable to find that kind of a growth direction. I mean, it must have been like a scary situation to be in as a senior leader there. I guess I'd love to get your perspective on, well, I don't know maybe Netflix is the next Yahoo kind of situation where you have something that looks incredibly defensible and immune, and now Netflix just has so many competitors and the technology has kind of been become commodified. It's hard to know where the company can turn since it hasn't really branched out into anything else to its credit. I guess I'd love to get your perspective on learnings from your experience, your 14 years of Yahoo, and how that resonates in terms of what you see in the market today.

[00:36:33] TT: Well, first of all, I think so many people that have left Yahoo, to go on to do really remarkable things. I mean, companies like Slack and WhatsApp, and Jeff Weiner is a CEO of LinkedIn, and Dan Rosensweig was at Chegg. There's just so many folks out there. So, it's just a really unique place and I think we always hired really, really sharp people. The culture always allowed you to tackle the gnarliest, hardest problems with complete freedom, right? It was all based on FreeBSD. We would build our own messaging protocols. I mean, everything was sort of built inhouse and it's just really fantastic time. And obviously, MapReduce, and all the Hadoop stuff came out of there, and what an incredible place.

I think what you're asking is, what could we have done differently perhaps? I think there's a few things. I think, again, you can get into this sort of like Innovators Dilemma problem pretty quickly, right? And you can almost become a victim of your own success and you always have to continue building your subsequent acts. Startups always talk about what's your second act and your third act, and so on. So, I think there's a few sort of really large themes we missed, and then maybe a couple of technical things, I would have perhaps done differently that would have given us more velocity and I didn't have control over that particular problem. But we'll get into that in a second.

The first thing we missed is, I think the content that we had on the sites became sort of commoditized very quickly, as you mentioned. So, we certainly reached a point, we didn't necessarily have to go to

Yahoo to get some of the news that you needed, because it was showing up and being syndicated everywhere. A lot of that was driven by social media. So, I think, not sort of grabbing social media by the horns and taking control of that was a pretty big miss. I think the other thing before that was obviously search. We had search, we were well positioned with search, there were some fantastic patents that we had, from companies that were acquired. There was fantastic search technology that was done. But Google built a better mousetrap with their search capabilities and I know there was ways to handle that perhaps more elegantly is sort of the best way I could get into it. And I think continuing to try to build your own search engine, when there's one that's already superior is hard, because it requires a massive number of developers to do it well. I think there's just there's different path that could have been taken with respect to both search and social media. And then I would also add, later stages, Yahoo mobile. It was just too late, the mobile. Should have gone the native apps, should have prioritized mobile over desktop, sooner rather than later. It was just a little bit too late by the time we started to really focus on it.

On the tech side, definitely I got there in 2003. I say up to about 2010 or so, you know, we had some of the biggest data centers in the world. It was all based on FreeBSD, amazing libraries, amazing packet systems to use really powerful software that just scaled incredibly well. But then cloud came. What did the cloud really give you at that time, circa say 2010? What it gave you as development velocity, because you could start to build solutions much more quickly, because you could start to glue services together, like I keep talking about, and we didn't embrace the cloud as quickly as we should have. I mean, obviously, all the web servers and stuff that's cloud. But when I mean, embrace the cloud, what I mean is using cloud services and third-party services, to build our applications. That idea of doing that and getting away from privately hosted data centers, running FreeBSD, which I would definitely say lost a lot of mindshare to Linux at that time, there's just less software you could use natively within FreeBSD.

We would have proprietary compiled versions of it and then it would start to deviate from what's an open source, because it's tuned for FreeBSD. All of that, I think really hurt our development velocity. We were just held on to this idea that we could do it in the data center using our proprietary tech. We held on to that too long, because the competitors gained access to similar tech, and it was provided to you by the cloud service providers. So, what suddenly happened is the playing field got leveled out on the technical advantage side, and suddenly, we didn't have that leverage anymore. I think that really hurt us, right? So, smaller companies can move faster, and they could develop solutions and iterate more

quickly than we could, say, at a larger company and that's always going to be the case. It wasn't specific to Yahoo.

So, that combination of missing really big themes that evolved over the years plus, I think the underlying infrastructure and tools available to developers really slowed it down.

[00:41:09] JM: As we begin to wrap up, I guess I'd like to ask about – maybe you could just wrap up by talking about the competition for infrastructure deals. I think, being an infrastructure investor, there's really not a ton of amazing infrastructure deals, I mean, from what I can tell. And the ones that are there, a lot of times, they're almost like obvious bets. But I mean, a lot of the knock out of the park at this point, bets are somewhat obvious. It becomes – because the team can be so strong, the engineering breakthroughs can be so obvious, and it really comes down to a matter of price. At least in the early stages, in the A and even in the B at this point. I just like to know about the nature of competing for a really good deal.

[00:42:08] TT: Infrastructure is a hot market. There are some really amazing outcomes, when you look no further than, say, Datadog or snowflake. I mean, these are just massive financial outcomes for their investors and employees. So, as you're saying, it's going to be pretty attractive for a venture capitalist to want to invest in, right?

The way I approach it is, it's highly competitive, but it's exciting. Frankly, I feel like I'm taking a pretty unique angle at the problem. There's not a lot of like x engineering leaders turned venture capitalists/ investors out there. And that's what I'm good at. That's my trick. I'm not the financial wizard that a lot of folks are. I'm not the most well-connected person. I'm pretty connected. But I'm not the best. There's always better someone at everything you can do. But who I am is a pretty good product person, I think, and I think I'm a really good engineer, frankly. So, I think the angle that can take with folks is talk to founders, developer the developer, and just ideate on the product and talk about what's not working, what's working well, what's not working well, where we could take it, sort of Crispin the vision and just ideate together, and there's a lot of folks out there that find that really refreshing.

They're excited to talk to an investor who can really, really talk shop at the lowest, lowest level. Let's take the Julia investment, for example. I was just really compelled by the idea that they're using LLVM under the hood, and how that's accelerating things and the pros and cons behind it. How hard is it to

integrate with the LLVM community and having that kind of debate and conversation with founders, it builds relationships really, really quickly. Same thing happened with my Pine Cone investment. I mean, Pine Cone is an embedding model vector database. You're carrying high-dimensional, large vectors for similarity anomalies, and things of that nature. But there's this whole ecosystem of ideas that can be built on upon that, applications on top of it, and the same way that's Splunk had, and now Snowflake is starting to have, that can leverage that power of vector search.

And when you can start to have that kind of conversation with the founder, they want to work with you, right? Because they're getting a lot more out of the conversation than say, an investor that has more of a financial background, for example. I can file bugs, and I'm going to kick the snot out of your API's and give you feedback and tell you what you think you're doing well, and you're not doing well. I'm doing that right now to a company I'm looking at and I believe the founder finds that quite refreshing, and personally, I find that really, really gratifying because it helps me deal with my scratching of the itch of wanting to code all the time and I'm actually going to take that one step further. I'm about to open source, a Pine Cone API written in Julia. So, I'm going to open source some software that actually connects two of the companies together.

So, I think, the founders I work with, they really liked that because you I'm having the conversations with him and I can ideate, but I'm also enabling them by opening open sourcing software for them. I think that model works. It's not going to work for everybody, but I think there's some founders out there who will be attracted to that and I'll be excited to work with those folks.

[00:45:14] JM: Cool. Well, Tim, thanks for coming on the show. It's been a pleasure.

[00:45:16] TT: Thanks. I'm really appreciative of you having me on the show. It’s fun.

[END]