EPISODE 1028

[INTRODUCTION]

[0:00:00.3] JM: Software investing requires a deep understanding of the market and an ability to predict what changes might occur in the near future. At the level of core infrastructure, software investing is particularly difficult. Databases virtualization and large-scale data processing tools are all complicated, highly competitive areas. As the software world has matured, it has become apparent just how big these infrastructure companies can become.

Consequently, the opportunities to invest in these infrastructure companies have become highly competitive. When a venture capital fund invests into an infrastructure company, the fund will then help the infrastructure company bring their product to market. This involves figuring out the product design, the sales strategy and the hiring roadmap. A strong investor will be able to give insight into all of these different facets of building a software company.

Vivek Saraswat is a venture investor with Mayfield, a venture fund that focuses on early to growth stage investments. Vivek joins the show to talk about his experience at AWS, Docker and Mayfield, where he currently works. He also talks about broad lessons around how to build infrastructure companies today. It was an enjoyable show and I enjoyed talking to Vivek as I always do. I hope you enjoy it as well.

[SPONSOR MESSAGE]

[0:01:32.5] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust.

Whether you are a new company building your first product like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals. Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer.

We've also done several shows with the people who run G2i, Gabe Greenberg and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack and you can go to softwareengineeringdaily.com/g2i to learn more about G2i. Thank you to G2i for being a great supporter of Software Engineering Daily, both as listeners and also as people who have contributed code that have helped me out in my projects.

If you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW]

[0:03:21.9] JM: Vivek Saraswat, welcome to Software Engineering Daily.

[0:03:24.3] VS: Glad to be here. Thanks very much, Jeff.

[0:03:26.6] JM: You've worked at several infrastructure companies. In 2013, you were at AWS working on Elastic Block Storage. Like me, you worked there less than a year. Tell me about your experience at Amazon.

[0:03:39.7] VS: Yeah. In many ways, I really enjoyed working at Amazon. I was on the EBS team and specifically, I worked on snapshots and on scheduling for snapshots. The thing that I enjoyed most about Amazon, I was up in Seattle is just how strongly Amazon believes in its culture, number one.

If you look at Amazon, Amazon has a set of leadership principles and Amazon believes very, very heavily in those leadership principles, to the point that when you're interviewing at Amazon the questions that are asked reflect those leadership principles and people look for those

leadership principles in your answers as a candidate. That's something you probably remember from your time in Amazon as well.
When you're in meetings, people really look at those leadership principles, things like bias for action and how they act. I think a lot of companies have values, but those values are often just there. They're somewhere on the website, or somewhere in the employee handbook. People don't necessarily live by them. Amazon's leadership principles are actionable and there's something that people use on a daily basis.

When I'm talking with my founders, I always bring back those leadership principles as an example as filling to use for their values. Something to live by and something for their employees to actually use, so that's number one. The other thing that I really respect about Amazon is just the laser-sharp focus on being customer first. I mean, everything is about making customers more successful and more effective. Whether it's as a product person, it's really this focus on the looking backwards principle.

When you're creating a new product, you start with where's the vision of we know, what is this product at scale from a customer perspective? You give an example of what an effective and customer looks like when the product is finished and working at scale and then working backwards to what is the MVP of that product. Everything takes that approach of what the customers approach sees and the product should be. Then everything you look at are the customer metrics around that product, the customer use case around that products. Again, it's always about making the customer successful. I think those are the things that I respected most about the company and those are the pieces that I take away the most.

[0:05:45.0] JM: Do you have any sense for how the AWS side compares to the marketplace side? I worked on the marketplace side.

[0:05:52.6] VS: Yeah. I mean, now it's been gosh, seven years almost. It's 2020 already, huh, since I've been there. I know that things have changed quite a bit. Back at that time when medium market was everything in their retail marketplace, or you're talking to that –

[0:06:07.7] JM: Just non-AWS.

[0:06:08.6] VS: Yeah. There was definitely a bit of a divide in culture back then that AWS was more of the typical IT tech culture in many ways. It brought in a lot of folks who came from the tech IT world, but still instituted a lot of the Amazon culture. AWS was a little insulated from the Amazon mothership in the way it was running in some ways.

I think that AWS is more integrated in Amazon than it used to be nowadays, particularly as a much, much, much bigger organization. I think in the earlier days, that's what it was like at the time.

[0:06:44.8] JM: After AWS, you were at VMware. Do you have a sense for how VMware responded to the market for cloud computing?

[0:06:54.1] VS: It's an interesting question. I think very, very early days VMware was very focused on the on-prem market and maybe didn't – and initially hadn't seen the transition to cloud as quickly as AWS clearly saw those with cloud native market. VMware did see that. Actually, what VMware saw was the focus on hybrid clouds. VMware created what was first called vCHS, vCloud Hybrid Service and then what was known as vCloud Air, which is how you basically make a service that runs in the cloud, but then allows you to shift applications between your on-prem service in the cloud. It was almost too ahead of its time.

Now you look and you see AWS Outpost, you see Azure Stack, you see Google Anthos and you see those services are pretty core today. This is 2014, I think when vCloud Air was there and it was almost too early. I think that VMware saw the cloud and at the time, hadn't responded as quickly as AWS and Azure and then Google, I built.

To be fair, I don't think a lot of people saw just how quickly and how big a giant the clouds were to become and the investment you needed to really go and be a cloud giant. It took a large amount of investment just to compete in those markets. VMware has innovated though and has built some pretty amazing things in the meantime and some amazing services in the form of software-defined storage [inaudible 0:08:13.8] and software-defined networking. I'd certainly wouldn't count them out. They've built some incredible innovations.

I enjoyed my time at VMware quite extensively. I was there working first on storage and then working on right around when I was there in the first 2014 and 2015, was right when containers started to become really big and saw – for me, I saw that this would become – I had the potential to become the new standard for next generation applications. Started finding the like-minded people who believed in then telling that to anybody else who would listen, and I formed VMware's cloud native storage applications product initiative. We started working on making persistent data work for containers back in those early days and started building some really interesting innovations around that area.

[0:08:57.8] JM: How closely have you been following the consolidation of VMware and Pivotal and Dell?

[0:09:04.7] VS: Yeah. I mean, I was there when Dell made initial acquisition of EMC and VMware before I moved on to Docker. Then since I left, I've been following some of that acquisition. I haven't been as close to what's been happening with Pivotal and VMware. I know that I think one of the big things that's happening internally, or that I see with VMware is just with the focus on DevOps, the market is moving closer to being organizations needing to be closer to developers. I think the consolidation of Pivotal and VMware has helped bring more developer DNA within the company. I think that's going to be good; really, really good for the culture of the organization.

I think earlier, VMware was very developer-focused and then shifted more to an enterprise IT structural organization. We saw that within the cloud native group at VMware, this need to focus more developers. I think Pivotal help bring in some of that talent within the organization. Now I think that consolidation really helps bring that together.

[0:10:03.1] JM: When you were at VMware, I believe that the product roadmap was impacted directionally by OpenStack. Do you have a sense for why OpenStack did not become the thriving Kubernetes ecosystem that we have today?

[0:10:19.8] VS: I think this is one of the biggest debates of the cloud native world over the last couple of years; why OpenStack went a certain way. Whereas, Kubernetes is growing quite heavily and we'll see how things go. Certainly, the signs are very, very positive. It's a very thriving ecosystem, right? Why one thing seems to have gone a certain way and one thing seems to have gone another? I've heard a lot of people talk about this.

I think my take is that there's two reasons; one is that OpenStack had a lot of cooks in the kitchen very early on. You can say that the core primitives of OpenStack never really had the chance to be built out to be successful, because there are so many cooks in the kitchen, so many vendors trying to make their individual implementations of the primitives. The compute storage, networking, etc., their vendor implementations of those primitives is successful.

You have so many vendors pushing it in different directions that the core technologies never really had the chance to be truly successful. That's number one. Number two and I always try to think about use cases and value propositions. I think when OpenStack came about, it never was really clear what the true value proposition of OpenStack was. Was the desire to build AWS on-prem? Was the desire to build a vSphere competitor in a world where most people were happy with using their vSphere software at the time?

You look at Kubernetes in comparison. Well, actually let's take a step back. You look at containers first. The first used case of containers is actually pretty clear. It was application and component versioning. It was actually the very first use case. That took off like crazy. Developers love that use case. Then everything around microservices and the cloud native world exploded.

You look at what Kubernetes is doing, it becomes application and cluster management and orchestration for next generation applications and that has exploded as well, in a world where multi-cloud has taken off, hybrid cloud has taken off. There's a right time, right place. I think most importantly, to the question of cooks in the kitchen, early on in Kubernetes lifecycle, there was a very strong player that was helping to govern Kubernetes in the form of Google, and then made sure that the core primitives of Kubernetes were strong. The core infrastructure within Kubernetes was strong. Then various other folks have come in and helped to build out the ecosystem around Kubernetes. Internally, I think the structure of Kubernetes is very strong and that's what I think helped to give Kubernetes the fighting chance that it needed to be successful.

[0:12:45.5] JM: VMware is a great example of a company that built a core infrastructure product for enterprises and then managed to keep a heavy presence within enterprises and sell more and more products. What did you learn about the process of market expansion from your time at VMware?

[0:13:05.5] VS: That's an interesting question. I think there's a couple of thoughts. There's expansion within a customer and there's market expansion. Those are actually two different questions; and one is tactical and one strategic. Let's talk about the strategic question of market expansion. You take a product like vSphere, vSphere has served these set of use cases around application management and orchestration and it's served that very successfully and generated billions of dollars of revenue for the company.

The use case was quite simple. With virtualization, you could take a server, you could dice it up and you could serve far more applications than you used to be able to serve. That maybe hit a certain wall within a customer, or within a set of customers. Talk about market expansion, VMware create – with a set of increasing technical innovations, created two interesting market expansion opportunities in the form of software-defined storage and software-defined networking. One of those was a major acquisition in the form of Nicira, which then became NSX, that's software-defined networking. The other was more internal. There were some hiring acquisitions, but the product was internal in the form of vSAN, which is a software-defined storage.

Ultimately, those are market expansions. Those provided brand new territories of expansion within the market, but that allowed to expand revenue within individual customers. Within the same customer base, you could provide brand-new products. That's one of the example of how they created brand new market expansion opportunities within the customer.

Tactically and this is probably relevant to startups is how you think about customer expansion. As you're expanding within a customer, you want to think about opportunities about how you can land an initial deal within that customer, find a use case that's really resonating with them, land that initial POC at a certain price and then expand within that customer, either through the same use case, but expand the amount of usage you're getting out of that customer, land within additional sites and additional usage within that same site. Or expand additional use cases, so additional products, additional areas. That's something more tactical at a smaller company.

A smaller company, you may not have three product lines. You probably only have one product line when you're just getting started. Being able to think along that same lines, how can you land an initial deal and how you can even expand? That's pretty critical in almost any enterprise business.

[0:15:21.7] JM: After VMware from 2015 to 2018, you were at Docker. Three years at Docker during the container orchestration wars, you saw a lot of blood, I'm sure. A lot of blood was shed in the container orchestration wars. What was it like being inside Docker in those years and seeing just so many people who were investing in their own container orchestration tools? Enterprises seemed like they were not yet ready to adopt container orchestration yet, because they were waiting for the market to consolidate around a particular container orchestration solution. What's your recollection of the container orchestration wars?

[0:16:11.0] VS: Yeah. I mean, I think it's interesting. Like in hindsight is often 20/20. People say," Hey, look. We can see where the world is now so very clearly." Back then, it really wasn't clear what was going to succeed or what was going to win. As a product manager, my job was to make customers successful. I spend my time understanding and again as a product manager, your job is really to have empathy for your customers.

You mentioned that customers are waiting and looking for – customers weren't actually really waiting. They were in pain, right? They were looking for solutions to solve their problems of how they would manage their applications at scale. That was where customers were for them they were giving us a set of requirements around how do we manage applications in a distributed fashion. In particular, you look at the buyer and they were IT folks and they were trying to help their developers succeed at building in managing those applications.

For us, it started with okay, how do we build them in a way – in particular, in the enterprise world, they were helping new developers get to cloud native applications quickly, who didn't already understand how to use applications. This is an interesting insight, I think. Today, it's easy when you go into the Silicon Valley world and you understand how to build microservices and cloud native apps. If you don't understand how to build microservices, 12-factor apps, etc., it's really, really hard to actually do that successfully.

In fact, I mean, just as an aside, as an investor, that's one of the things that I'm looking for right now in terms of startups is people that make it easy to help developers build like microservices and cloud native applications. That's still very much an unsolved problem, I think in the world of Kubernetes and in cloud native apps.

Going back to the point though, this is something that enterprises were in pain and trying to solve for. I think that's one of the things that the team was solving for on the open source side with swarm was in building a simple way to get containers and next generation applications running.

[0:18:15.8] JM: The Ruby on Rails of container orchestration.

[0:18:18.4] VS: That's an interesting way to put it, I think in many ways. It's give people a simple way to get things done at the sacrifice assemble of level of complexity. Ultimately though on the enterprise side of it, of the product, my job is to give people what they want, right? We built the product, we gave folks a product that consisted of open source swarm, plus several integrations on security, CICD integrations with networking, storage, etc., and a pipeline all the way from the developer through to the IT operations manager managing the application at scale.

Then eventually, folks were starting to use Kubernetes at scale and we added Kubernetes to the enterprise product as well. We were actually running Swarm and Kube side by side on the same cluster, and you could choose which orchestrator you wanted to use. This is actually in the same cluster. You can choose which orchestrator you want to use. Now we’re actually running three orchestrators side-by-side; Swarm Classic, Swarm Kit and Kubernetes, which was a testament to the skills of the engineering team at Docker.

Long story short, it wasn't clear at the time what people wanted or what people were going to use. We were just reacting to market needs at the time and building the product that people wanted to see.

[0:19:28.5] JM: Why did Kubernetes win?

[0:19:30.8] VS: Yeah. I think that's an interesting question as well. I think it comes down to the ecosystem, right? I think there's a strong ecosystem that's been built around Kubernetes and it's been incredibly successful. I think ultimately, when you look at anything within the world of open source and within infrastructure, ecosystem drives everything, right?

One advice to anyone building companies out there is you have to build a thriving ecosystem, even with actually closed source products, ecosystem also drives everything. Take a look at even products like Slack, they have thriving integrations across a number of different products. If you look at products that are built across integrations, like Zapier for example, it's all about building integrations. You look at products like Octo, it's all about building integrations.

I think how you build your ecosystem in the enterprise role is extremely important. I think Kubernetes is no exception. It’s build an incredible ecosystem. That ecosystem is composed of both software integrations, as well as people. I was recently at KubeCon back in the fall and there was what? 13,000 people at KubeCon, from 8,000 a year before. It's just incredible. It's incredible to see that energy. Hats off to the teams you have and the folks for building that amazing ecosystem.

[0:20:43.7] JM: What about Mesos? Why didn’t Mesos win? I mean, Mesos had community adoption, enterprise adoption.

[0:20:51.3] VS: I mean, I think again, it comes down to well, a combination of things. You could argue that each of the orchestrators had their use cases, have their use cases. Mesos is still used by players in various places. Swarm is still used by players in various places. Kubernetes is used. Nomad is used. Kubernetes is probably the most flexible of orchestrators, since the use across the widest set of use cases, but it does have the strongest overall ecosystem. It had a set of strong players who helped guiding it. It has a good – the CNCF has done a great job in backing and guiding Kubernetes as well. I think that's played a hand in it being quite successful.

[SPONSOR MESSAGE]

[0:21:37.9] JM: When you start a business, you don't have much revenue. There isn't much accounting to manage. As your business grows, your number of customers grows. It becomes harder to track your numbers. If you don't know your numbers, you don't know your business.

NetSuite is a cloud business system that saves you time and gets you organized. As your business grows, you need to start doing invoicing and accounting and customer relationship management. NetSuite is a complete business management software platform that handles sales, financing and accounting and orders and HR. NetSuite gives you visibility into your business, helping you to control and grow your business.

NetSuite is offering a free guide, Seven Key Strategies to Grow Your Profits at NetSuite.com/sedaily. That's NetSuite.com/sedaily. You can get a free guide on the Seven Key Strategies to Grow Your Profits.

As your business grows, it can feel overwhelming. I know this from experience. You have too many systems to manage. You've got spreadsheets and accounting documents and invoices and many other things. That's why NetSuite brings these different business systems together.

To learn how to get organized and get your free guide to Seven Key Strategies to Grow Your Profits, go to NetSuite.com/sedaily. That's NetSuite, N-E-T-S-U-I-T-E.com/sedaily.

[INTERVIEW CONTINUED]

[0:23:26.4] JM: The container orchestration wars were an indication that companies have a strong desire to go through a re-platforming. There's a re-platforming, or modernization that companies are going through at this point constantly, its modernization. One way of organizing this within a company is by defining a central platform engineering team. If you were managing a platform engineering team at an average enterprise, what would be the baseline level of value that a tool would have to provide in order for you to allocate engineering resources?

Because any given investment in platform engineering is a trade-off against business logic, against logic that's actually going to drive your product forward in a way that is going to impact the customer. How do you organizationally trade-off between resources allocated towards platform engineering, versus business logic?

[0:24:45.7] VS: Yeah. The interesting thing is I think you just almost answered the question right there. This is something that I actually hear a lot about talking with various engineering and DevOps leaders. I think it's incredibly relevant to anyone building an enterprise infrastructure startup today.

If you're building an enterprise infrastructure startup today and you're building a bottoms-up-based, or product let-go to market business base, your users might be developers. Your buyers are going to be someone most likely up the stack. That buyer is probably going to be either somebody in a platform engineering team, or a DevOps IT team that's effectively fulfilling a similar type of role, or an enterprise architect team that's building the platform that developers are going to use.

Understanding the value proposition that that platform and/or IT platform team is looking for is going to be extremely important in the sale that you're trying to make in creating a customer. In talking to these teams, I think one of the biggest pain points that's actually come out of the cloud native world today is how much focus has been shifted away from business logic. If you look at the CNCF landscape today, the last count that I looked at, there's one 1,287 tools I think it was in the CNCF landscape. That's incredible.

I mean, it's incredible A, from the amount of innovation out there. It's also incredible, because if you're an engineer who wants to go and build code that serves a purpose for your company, the amount of tools that you have to integrate to get the job done is in order to actually focus on building what you want to build, it just drains away from the time that you actually need to do to focus on business logic. That's just talking about back-end and front-end tools, you actually need to solve a problem. That doesn't even consider the tools that you need to do for workflow and processes.

Whether it's working in github for checking in your code, whether it's working in developers and things like Figma or Envision, or whether it's working in JIRA, or tools like that from a project management standpoint. Or actually, I even forgot. If you're working in remote tools, you're working with things like Slack, or Zoom in order to just get meetings done. Just the amount of things you actually have to do before you can actually write code to a focus on business logic is ridiculous in today's world.

Now go back to that platform engineering team and put yourself in their heads. What are they trying to actually accomplish if they are trying to make their teams of developers on the other side successful? The main thing that they are now trying to think about is how can I make my team's focus the most on what they need to do, which is on business logic?

My take there is that the most successful next generation of tools are the ones that allow developers to actually focus most on business logic. It's actually abstracting away the levels of complexity, of tools and interfaces as much as possible. I have this theory around what I call developer augmentation, which is how can you let developers focus the least on integrations between tool sets? How can you stitch together sets of tools, automate repetitive tasks, so developers can focus on key workflows?

That's what I would say to anybody who's building new tools today. Focus on how – If you're going to sell to that platform engineering team, the value proposition you can think of is how can those developers that you're actually building as your users, how can they make it fire-and-forget, right? How can they actually build on that business logic? That's what I would err towards.

I think the previous generation in cloud native as we were building the primitives of the cloud native world was around complexity, because we were building that first layer of tools to make cloud native successful. That was day one for cloud native. We've hit cloud native day two, where we actually need to operationalize this technology in production. That means I think from a developer standpoint, you need to make developers successful, so that they can focus on business logic. That means abstracting away that complexity.

[0:28:43.5] JM: Can you give any examples of companies that fit that bill?

[0:28:45.8] VS: There's a bunch of companies on the engineering effectiveness side that are just starting to hit the workflows. There's companies like UPLevel and Pinpoint that are focused on providing metrics for engineers. I think there's companies that have been around for a little while, like if, this, then that, that have been about stitching together workflows are Zapier, that have been about integrating APIs between across each other. I think those are early examples of that trend.

The other interesting example this by the way, is there's been a lot of focus on low code and no code tools. A lot of people have been focused on low code and no code tools as a way of bringing people who don't know how to code into the coding world, which I think is fair. Actually, talking to a lot of engineering folks, a lot of people I know are engineers who use low code tools as a way to do tasks that they don't really want to spend a lot of time on.

[0:29:37.2] JM: Really?

[0:29:37.5] VS: Yeah. It's a way of automating certain sets of tasks effectively in a quick manner, so they can focus on higher level work, which –

[0:29:45.4] JM: Have you really had a substantial number of con –

[0:29:47.2] VS: Yes.

[0:29:47.8] JM: Really.

[0:29:48.1] VS: Which I thought was not no code, mind you, but low code tools. Not drag-and-drop interfaces, but more low code, which I thought was really interesting.

[0:29:55.4] JM: What is that work? Can you give me an anecdote? Some engineer you talked to that used some low code thing to do X.

[0:30:01.2] VS: I mean, there's so many low code tools out there. I don't know if I could give a ton of examples. I mean, there's so many companies out there. You might have heard of Dark, for example. I think you had –

[0:30:10.4] JM: Yeah, yeah.

[0:30:10.8] VS: - them on your show. That's an example of a tool that I've heard too, that's been using as better to create examples. Companies like [inaudible 0:30:16.8] in this example. We made an investment in companies using this example. There's a ton of different companies within the low code world that have been used for this. I actually think it's super fascinating, because it's not something I would have expected.

[0:30:32.0] JM: I mean, I agree with you it makes sense. I mean, I just talked to the Parabola founder. What I just wonder is –

[0:30:40.6] VS: You've heard of Retool, or –

[0:30:42.9] JM: Sure. Yeah.

[0:30:43.9] VS: There's a couple other companies within this internal tooling space. It's actually a really good example of this use case. These companies are all focused around building internal tooling really quickly. The idea is internal tools usually are pretty simple, right? They're usually a series of quick spreadsheets or databases with a UI built on top. They're not anything super complicated, but they need to be done, right? Why would you spend a ton of engineering resources to build on that scale if you could use a low code tool to get that done quickly? That's a good example of this use case.

[0:31:15.5] JM: Right. I understand this philosophically, hypothetically, rhetorically. But realistically, I want to talk to the engineers – I guess, I need to talk, find some customer use cases or something, because what I'm trying to understand is I get it that these things are most likely the future. The question is when is that future arriving? When are there going to be enough people that are building internal tools five, 10 years? Is it going to be five or 10 years out, or is it tomorrow? I get it, that spreadsheet workflows are bad and we should probably be working at a higher level of abstraction. Do the people who are working with spreadsheets, those are the people who are voting with their feet. Are they actually going to be adopting low code tools?

[0:32:00.8] VS: I mean, having worked in a couple of different enterprise companies and startups, I mean, I know engineers who have built these kinds of internal tools and these workflows. It sucks, right? I've seen the use cases myself and having talked to enough leaders, I know that things lie these are a problem.

It's not just internal tools. There are other use cases I've heard of. I will say that as applications continue to get more complex and more distributed, there's enough low-level work that's being – many of that has the potential to be automated further, I think. I think the analogy you made around Ruby on Rails is an interesting one. Well ultimately, what did Ruby on Rails do? It allowed you to quickly construct together an app, a web app.

Some people criticize that hey, it put together a magic that shouldn't have been put together and it abstracts away things that you shouldn't abstract. I mean, yeah, you could be encoding in assembly I guess at a certain point. I think as applications become further and further more distributed, you're going to be doing the same thing. You're going to be automating key pieces of tasks. That's the bet that you're making with these tools.
I think that's the use case of low code, or these kinds of toolings. Again, I think low code is just an example and just a nomenclature. 'm looking more broadly at what are the kinds of tools that can be used to automate repetitive tasks within the engineering and product development world. This would apply for a UX and for product managers too and there's a lot of repetitive tasks across the product development world. How can you automate that and allow people to focus on business logic? Back to your initial question of what would a platform engineering or a DevOps manager or VP really look for as a value prop.

[0:33:40.4] JM: Yeah. As an investor though, let's say cool-looking low code tool comes into your office and gives you a pitch and they say,“ Look, we've got five customers that are startups in Silicon Valley and we have one POC that's a manufacturing company in the Midwest that uses us for workflows.” My question is how do you figure out whether that's enough to extrapolate to being an actual, significant market that can be expanded? Are people ready to adopt this? I guess, my question would be how do you size the market that's ready to get these tools?

[0:34:25.2] VS: Yeah. That's a really good question. I mean, to take a step back for a second, the way I evaluate investments is around four distinct areas in this order. It’s people, potential, product and progress. Ultimately, any investment is around the people themselves and an evaluation the people that's at the early stage where I invest seed Series A, the people are the most important thing and they're the ones who are going to take it. That's where I'm spending the vast majority of my time.

The second one is the potential and how big can this really get. Market sizing is an interesting question. There's a reason I call it potential and not purely market sizing, because it may not be a big market now, but the market could expand right. It may be a big market, but it might be constrained in various ways and you have to think through a lot of those angles.

You can look at a market top-down, or bottom-up. Top-down is you go look at some analyst report, IDC Gartner. It says the market has this many billions and we are this 10% of this market and we'll take that bite and that's how big we are. Usually, that tends not to be in my view a very good way to approach the market, unless it's a really traditional market, which most of our investments typically aren't. Maybe at a later stage it is. At the early stage, that's usually not the case.
Even in not an early stage, if you looked at say a company like Uber, or Lyft, one of our investments at the series A, back when it came out and you're like, the on-demand car market is as big, you probably wouldn't have made that investment, right? Same thing like the app, like the Kubernetes market is as big and back when we invested in Rancher back in the day. You wouldn't have said that there was that market. What do you do then? Well, you look at what I call the bottoms-up approach, which is taking a parallel of how many – the type of buyer that could eventually build this market at scale, how many of those buyers are there at scale. Then as this market grows, how can you expand that market and work with us have the potential to be.

Let's take your example of this theoretical internal tooling market. This company may have one customer now. Let's say you believe that there is going from user to buyer that the buyer for this internal tooling market is the platform internal, the internal platform engineering tool. I'm not saying that it actually is, but let's say for this theoretical argument that it's a platform engineering tool within a more traditional company that doesn't necessarily have a strong engineering team to go build tools; theoretical argument.

Then you can go and say,“ Okay, how many of these companies are there out there within say, the US?” There are X number of companies. There are a 1,000 companies. There are 10,000 companies. Then you can start making some theoretical estimates of what are the rough deal sizes that each of those companies could pay you. Is it a 1,000 companies that'll pay you a 100K annual contracts roughly, or annual recurring revenue?

If you can get a 1,000 companies pay you a 100K a year, that's a – I'm not too good at math, unfortunately as an investor that's a bad thing, but pay you a 100 million dollars a year, right? That's a large-scale company. That's something that’s interesting adventure scale. That's the things that I'm trying to think through when I'm looking at an investment. Is it a 1,000 companies paying you a 100k a year? Is it 10,000 companies paying you 10K a year? Is it a 100,000 companies paying you 1K a year? What is the scale that it looks like we're doing a bottoms-up sizing? That's how you think through something potential.

Then next stop, you start looking at the product and assess the product. Someone might be able the product’s background. I can take a deeper look at that. Then the final thing you look at the early stage is the actual progress of the company. What have they achieved in terms of traction? Ultimately, if you're looking at the early stage, the early, early innings of the company, that's actually the least important thing. The others are the far more important part.

[0:38:11.1] JM: Let's talk through more of the tactics of building a infrastructure company. Talking about low code tools or internal tool building is not exactly where you spend most of your time, I think. I think you spend most your time more on the infrastructure side, a little bit lower level. In that domain, you're largely looking at companies that are what I might call, point solution providers. Basically, these are companies that are building infrastructure products that are not AWS, or Google Cloud, or Azure. They are companies that are not actual cloud providers. They're building a logging solution, or a database, or some particular tool within infrastructure. Tell me about the competitive dynamics between a point solution provider and AWS. As a point solution provider, how do you build enough of a product following to have a successful company?

[0:39:16.2] VS: Yeah. That's a really good question. I will answer that, but I'll give a little caveat in terms of what's a point solution. I think really, when you're looking at in venture, you're looking for companies that want to have big outcomes, either that are going to build really big companies, whether it's going for an IPO, or a big exit. That means it’s something that's building a small point solution and won’t necessarily be successful. I think most big companies are ultimately building platforms. It's not just a point solution solving a small problem.

I think ultimately, every large successful company is building a true platform. They're not necessarily building an AWS, like an entire cloud provider, but they're still building a platform. I think that's ultimately what provides that level of differentiation. We'll get there. I just wanted to make that distinction between what's a point solution, which is you're solving a single use case that does something specific. Then there's a continuum between that to a platform which solves a set of use cases and then you build an ecosystem and you build a sustained community and set of integrations and vendors, all the way to your AWS, Azure, GCP. I mean, you're building an entire cloud provider. There is a bit of a continuum there. Let's set that.

The question was then how do you go about and build a successful business that in the face of the cloud giants, which I think is every company in the infrastructure world is facing that. Ultimately, I think it comes to innovation. Every startup that thinks within this space, you have to build a product that ultimately out innovates. If you're going to build something that the cloud vendors can potentially pick up on, you need to out-innovate against the cloud vendor. That's the single truth within the startup infrastructure world. That can be done in a couple of different ways.

If you build an open source project that a cloud manager can go and build an open source project on top of, then you either need to have the expertise on top of that to go and build either core proprietary features, or a better host experience, or something else that allows you to provide innovative features on top of that that make you more successful. Or you need to build in an area where the cloud vendor doesn't necessarily have as much critical experience. For example, there haven't been as many security, or core monitoring features on the core – a lot of the cloud platforms. That's an example of something that there have been a ton of innovations within the startup world.

There haven't been as many cloud vendor-related platforms in workflow related companies. You might think of things like PagerDuty for example, or anything that works within how product development teams actually go about doing their business with and getting things done. Those are examples of how you can build innovation and proprietary IP. That's different than what the cloud vendors can do.

I think ultimately, it just comes down to out-innovating and out-executing. This is what a startup does best. You find a core value proposition, that's what you're good at. You out-innovate and execute. You build something there. Then again, once you focus on there and you build and build a core business and do it extremely well, once you've built that to a certain scale, this is on that continuum where you can go ahead and go out and build a platform.

What is a platform? A platform is where again, you've built out a set of use cases and you've built an ecosystem. That ecosystem consists of a set of dedicated users and it consists of integrations with the existing outside world. That outside world can include the cloud vendors. It'll include other vendors as well. That's how you build sustainability within your business. I think if you look at all the major infrastructure players out there that have been successful, they've done this really well. I think that's again, how you compete in this business.

[0:43:07.9] JM: The reasons that companies default to AWS, or point solutions, like the companies that go all-in on AWS, does it have to do more with IM policies and the fact that you can easily adopt AWS tools more easily than some product outside of the AWS ecosystem? If I'm all-in on AWS, it's really easy for me to glue stuff together with AWS lambda. Why would I go with CockroachDB when I can go with whatever the AWS equivalent is that has easy hooks for lambda and easy hooks for my IM policies? Why would I do anything other than go all-in on AWS, if I'm even slightly in on AWS?

Or is it more about – do people just trust AWS more and they trust, they know the name AWS more than they know the name CockroachDB? Why do people go all-in on AWS, rather than stitching together point solutions together with AWS?

[0:44:13.0] VS: I think this is why as a startup and as a founder, it's really, really important to understand who your target user is and actually really importantly, who your target user is not.

There's always going to be a community of folks who they're all in on the AWS platform for simplicity, right? They've started using AWS products. There's other AWS products that integrate with AWS products. To your point, if you've already integrated with IM as a company, if you're a small shop that is a single a single vendor that runs everything within AWS, within AWS availability zones, you know you're going to use AWS-related products, you have no desire to build across multiple clouds, you have no on-prem presence, it doesn't matter to you whether you're running other tools.

You go to Amazon's re:Invent, you see new AWS tools, they just work for you. You're probably the wrong audience for a startup that's talking about multi-cloud, or that's talking about other value propositions. As a startup, you need to recognize that that's not your core audience and that not going after that – that going after that audience is going to be a waste of your time.

Understanding who your core audience is and who your core audience is not is critically important. I think there's always going to be that audience that exists. Then finding the audience that fits your key value propositions, whether multi-cloud is the right value proposition or whether it's not, whether there's some other key value props that you're looking for, HIPAA compliance, for example and security, might be something that IM won't be the right value prop for, is something that you should be considering.

[SPONSOR MESSAGE]

[0:45:52.6] JM: Open source tooling is generally preferable to closed source tooling. Because with an open source tool, you're going to know what code is running, you're going to know what the community is saying about that code and you're going to have flexibility.

Scaling open source tools is not that easy. You're going to have to spend a lot of time managing and maintaining that open source software. The alternative is to use closed source software, which will be scalable, but you won't know exactly what code is running and you won't have easy migration path.

Logz.io is a scalable and fully managed observability platform for monitoring, troubleshooting and security and it's all based on open source tools, like the ELK stack and Grafana. Logz.io is open source software at the scale that you probably will need if you are a growing company.

Sign up for logz.io today by going to logz.io/sedaily and you can get a free t-shirt. You can go to L-O-G-Z.io/sedaily, create an account for logz.io and make a dashboard. Once you make that dashboard, you will get a free logz.io t-shirt. Thanks for listening to Software Engineering Daily and I hope you check out logz.io. That’s L-O-G-Z-.io/sedaily.

[SPONSOR MESSAGE]

[0:47:30.5] JM: Legacy enterprises have so much inertia when it comes to on-prem workflows. If you go to KubeCon, or you go to AWS re:Invent, you can have conversations with large enterprises that are adopting cloud slowly, but steadily. What I don't understand is how rapidly these legacy enterprises are adopting cloud and what kinds of workloads they are comfortable adopting and what are the kinds of workloads that are just fine to run on the data centers that they've already invested in?

[0:48:10.5] VS: Yeah, that's a really good question. I mean, the traditional answer used to be that so-called legacy enterprises, traditional enterprise organizations were uncomfortable running primary applications, OLTP, etc., in the cloud. At first, it was only SaaS applications, Salesforce and things like that, were the only things that would run in the cloud. Then it was you do backups in the cloud and then only secondary applications. Then it was only cloud native applications would be built in the cloud. Back when I started at Docker, this was the thing. All traditional existing applications would run on-prem and then only greenfield applications would run in the cloud.

Now what we're starting to hear, I spend a lot of time talking with CIOs and CTOs, we're starting to hear that workloads are being shifted to the cloud. Brownfield workloads are being shifted to the cloud for cost reasons. A lot of traditional enterprises are under the gun to reduce CapEx and to reduce OpEx. There is an opportunity to move closer to the cloud. That's an opportunity also for startups for tools that help make it easy to tie their shift in migration, or to make – do policy management, to do security management, etc. That is an opportunity.

Though the caveat of that as you're thinking through building your GTM pipeline is that selling to traditional enterprise is a different ballgame than selling to mid-market, or selling to smaller companies. It's a longer sales cycle. Typically involves building a direct sales team and selling in a more top-down fashion. That's something to consider what kind of a GTM model are you really building. Early on in your company lifecycle, I would very, very strongly consider what kind of a company is it. If you're doing a bottoms-up model with inside sales, you can really build a fast velocity that you want if you're selling to global 2,000 customers immediately.

[0:50:02.7] JM: What are the biggest lessons we can learn from HashiCorp?

[0:50:05.1] VS: Good question. Yeah, so for those don't know where I'm at, we were the series A investors in Hashi. I think Hashi has done obviously incredibly well in the last several years. I think there's a couple of extremely key lessons that and working with Mitchell, Armon, Co-Founders, as well as Dave, the CEO, has been incredible.

There's some very key lessons. I'll start with one of the biggest things is the incredible balance and transparency between the community and the commercial. HashiCorp has built amazing community presence, but they've also – this has been shown across the github presence. It's been shown at the conferences with HashiConf. It's been shown with the dedication that Mitchell and Armon and the team have, both on Twitter and on github, in talking with the community.

They've also been very transparent on where they engage with the committee and where the commercial site is. A good example is that they made clear that much of the website is devoted to talking about the commercial product and the engagement in the community is primarily around github. They made very clear that that's where the dividing line is. There's no obfuscation that these two things are different. It's just very clear that that dividing line is there.

There's a commercial part that needs to be there to keep the business success – to keep the business running. HashiCorp serves the needs of the community and is dedicated to the community and that community is mostly served through github. That's where developers live for the most part, so that's okay.

Having that dual dedication, but having that transparency around the two has been incredibly successful. I think that's one. Two and this is one of the lessons that I actually bring to any founder that I talk with is despite having that transparency around the two, the teams around community and commercial are heavily integrated. There's no dividing line between community and commercial teams. The same product team that's making community decisions is making commercial decisions.

I think this is incredibly important. When you're building a company, don't have a community product manager, for example, and a commercial product manager. You should have one team and that is making the decisions, or a community product team and a commercial product team. Yeah, that's a better way to put it. Have one team that's making those decision.

The reason for this is clear. You don't want to have dividing roadmaps that are going in different directions. You don't want to have somebody that doesn't understand and have empathy for the needs of the community, or understands the community but doesn't have empathy for the enterprise, or vice versa.

You may have people with different expertises, but they should be on the same team and talking with each other regularly. You can have that cross-cultural DNA. I actually advise when you have a product team, people should have vertical specialization. Maybe they have – one person owns a product and then decides what goes in commercial versus community, free versus paid for example, within that product. I think Hashi has done that extremely well. That's one side within that.

I think the other really – so aside from that dedication on both the community and the commercial front and doing that in a transparent fashion and doing an integrative fashion, the other key thing that the company has done really well is in building an extremely, extremely effective go-to market, bottoms-up go to market pipeline. I think this is one of the most effective things within the company's success in going from startup to successful organization. Happy to expand on that, if that's helpful.

[0:53:32.6] JM: Yeah, sure. Talk in more detail.

[0:53:35.9] VS: Actually, it might help if I give an example of a specific product, I think. You're familiar with Vault?

[0:53:40.6] JM: Of course. Liquid management.

[0:53:42.5] VS: Yeah, exactly. For those who don't know, so Vault is a secrets management platform based on a key management store; effectively, the way then which most people do secrets management in distributed clusters today. Here's how Vault – Vault is a extremely successful open source and commercial product today. Here's what the go-to-market pipeline effectively for Vault looks like and what has made it successful.

Vault has an open source and a commercial component. Open source developers download and use Vault primarily from github and download and use it. It has a very what I call, critical path value proposition, right? You use it for managing your secrets. Secrets are important things. They could be your passwords to your database. It could be whatever one is your application. I use this term critical path value proposition. For me, that is something that is solves a fundamental pain point for a user, something that is so important that it is designed into the user's workflow.

In this case, open source users use it, but they can't exist without it. It gets used for say, one user and developer within their – on their local machine. That has become so popular that now within a company, every developer is now using it within their distributed clusters across the entire company. Now it's been designed into every cluster and workflow across the entire company. This has now become critical path. This has become something that's so important to the company that is critical architecture.

Now this is the key part. The IT organization comes in, looks at this and sees,“ Okay, there is this critical path piece that is so important to my architecture. I really don't have a choice but to support this.” Vault markets to this. Or HashiCorp, I should say, markets to this buyer and says, “Hey, this is a really important architecture. All your open source users are using this. Wouldn't you like to be able to support your developers and make them successful?” This I think is the second part of the key insight.

In a bottoms-up business, you're not typically selling to your developers. Developers typically don't buy the product directly. Importantly, I don't – the key insight here is I think your developers don't sell to your buyers. Developers don't sell to IT. Developers are passionate about products. Don't get me wrong, developers are extremely passionate products. But developers are not going to go to your IT folks and say,“ You need to buy this product.”

What you need to do as a company is find a reason to influence the buyer to offer the product on behalf of the user as a service. That is the key thing. A company goes in markets to the buyer, the IT folks and says,“ This is why you should offer Vault as a service to developers.” The buyer, the IT folks then come inbound to Hashi, offers it. Then the inside sales team at Hashi then takes that and offers a commercial version of Vault.

There's a two-fold part to this. Vault, then they can just drop in a license key and take the enterprise version of Vault, which is a proprietary – it's an open core model. It is both supported and there are additional features on top which helps sweeten the deal and make the sale that much easier. There's separate questions on whether you should have paid only support, versus an open core model. We can get into that discussion as to why you'd want to do open core, or just go paid support.

There's some lessons to be learned here. One is building something that is a critical path value proposition to the user. That's so effective that it virally spreads within the organization and becomes designed into the architecture in the workflow. The second is marketing to the buyer and influencing them to help them understand why they should offer this product as a commercial supported opportunity for the user. Why they should buy this product in order to help support the developer and the user.

Then making it trivially easy for the buyer to upsell, right? In the case of Vault, drop a license key, you have Vault Enterprise. It's an inside sale. There's no expensive field sales process. They just need to call in, or Zoom calls it an inside sales process. Now over time, now there's a top-down sales process, because Hashi is a big company. For years, they just had an inside sales model and it worked very effectively. This is how you build a product-led GTM bottoms-up adoption model. This is how Vault was able to build such incredible revenues over time.

[0:58:07.1] JM: This is a bottoms-up open source, or open core strategy.

[0:58:14.3] VS: Correct.

[0:58:15.0] JM: Basically, you have Vault as an open source piece. Random developers at a company pick it up, they start using it throughout the company, the product propagates throughout the company, gets used more and more. Over time, the company decides,“ Okay, we need to get the added enterprise features. We need to pay money for this thing.” You're pointing out that the transition from that open source tool to the paid tool should be very smooth. In this case, you pick up a phone, or take a Zoom call with a salesperson, you get a license key and you drop in the license key and walah, it becomes an enterprise distribution.

That's really smooth. It's reminiscent of the Slack model. Slack just propagating through a company and then eventually, a company saying, “Okay, we actually need to pay money for this, so that we can search through conversations, or get I don't know, other kinds of things.” We were talking before the show about Snyk, or Snyk, the S-N-Y-K security company, which takes another bottoms-up form in the sense that it's somewhere in between Slack and Vault, in the sense that it's an infrastructure product, but it's closed source.

If I recall, basically their free system is vulnerability scanning tool that you integrate into your CI pipeline. You just create this CI step where might as well scan your code through Snyk’s vulnerability database every time your CI pipeline runs. That's a great free insertion point. Once you get that free insertion point, there's all kinds of opportunities for expanding there. I think of that as another flavor of this bottoms-up model.

[1:00:06.8] VS: Yeah. By the way, Snyk has done incredibly well. They're one of the few examples of a bottoms-up security company. There's not many and they've done extremely well and raised a big round recently. Kudos to the Snyk team. You actually phrased the bottoms-up process quite beautifully. It makes it sound really simple. I think there's a couple of key speed bumps that I think people need to be aware of in this process.

The first is understanding A, have you really built a critical value path proposition for the initial user in the open source, or the free in the case of a product like Snyk? Is what you're doing actually critical path? Is it's solving a real fundamental value pain point for the user? It's surprising how many products don't. Just because it's open source, even if it has a lot of Docker pools, doesn't mean it's actually solving a fundamental value pain point, just because it's downloaded a lot. That's a key thing. You're familiar with the vitamins versus painkillers discussion? Is that something you've heard of?

[1:01:04.2] JM: Sure. Vitamin is something that makes you do a little bit better, but it doesn't necessarily – you're not begging for it. The painkiller is what you're begging for.

[1:01:12.8] VS: Yeah. I mean, like a vitamin you take every day, it might make you better. If you forget to take it, it's not a big deal if you forget to buy it from the pharmacy. You can always get it the next day, or the next month even. Painkiller is something you need now. It solves a problem you have right now. If you're not building a painkiller, you're probably not building a good business.

This is why having that empathy, even if you're not a former product manager and a startup, having that empathy for a user is so critically important, because you need to understand what that pain point is. This is why talking to your users and also buyers to different groups of people often is really important. You need to understand the pain points and the value props for each group.

Do you have that fundamental pain point right? Do you actually have it right? Are people actually using you for – and do you understand why they find you valuable, to the point that you're actually designed across the organization? Are you actually spreading virally? That's a good question to ask. Because an open source, you may not actually have the metrics that you – you may not have telemetry. In SaaS, you often do. In open source, you don't. How are you measuring your community correctly? Do you actually have that right?

Then the next critical speed bump I think most people get wrong is the upsell in two different forms. One is do you have enough value in your paid product? A lot of products fail, because there's too much value in the open source. I could get a lot of flak here for saying that in the open source versus paid discussion. The truth remains is if people see a lot of value in your open source product and decide that you're not worth paying for whatever reason, so be it, right?

There are some businesses that have done very well on paid support. Red Hat Linux is a good example from RL. Elastic did quite well for many years on paid support. Rancher has done well on within our portfolio on paid support. It can be done. To do that, you need to have a really good critical path value proposition that paid support is really important, that if this product fails, all hell will break loose at the company. You better be really, really confident in the value proposition – solving that fundamental pain point.

Then the other way is if that's not – even if that's not the case, building proprietary features is the other way in the enterprise product to ensure that enough sell matters. Are you building a strong enough enterprise product and continuing to innovate on the enterprise product in a way that drives people to upsell? I mean, for me, I think enterprise value is driven by a couple of different buckets.

I think everybody's first enterprise product is some combination of access control, or back, A back, etc., single sign-on, like LDAP, SAML and a pretty UI. That's everybody's one data enterprise product. That was my one data enterprise product. They think that's what will sell enterprise. Well, truth is no. That's what gets you on the table, that's table stakes. That might get you a couple deals. That is not what fundamentally drives enterprise value. You have to figure out what drives enterprise value.

For me, I think there's three buckets. There's performance, collaboration and peace of mind are the three primary buckets that drive enterprise value. There's individual features within those. That really depends on the business you're in, so I think that's a – that really depends on the area. You have to really understand what drives value, what are you willing to put in the free versus paid bucket and how are you constantly innovating? Because things will keep shifting into open source. How are you constantly innovating to make sure there's enough value that you upsell customers?
Then the last key point of that is how do you make the upsell process painless, right? I mentioned the point about dropping in a license key. You wouldn't believe how many open-source folks require you to switch binaries, for example, when you're upselling to the enterprise

product. Switching binaries is tough. Let's say you've got a 100 instances in enterprise customer. Are you going to switch binaries on each of those instances? That's a lot of work.

For a lot of enterprise customers would be – or a 1,000 instances of an on-prem product. You're going to be like, "That's not worth my time. I'll just keep what I have and support it." Drop in license key, or dropping – building your enterprise props in a way that just drops on type of the open source makes life a lot easier. Or if you're a SaaS-based product, this isn't even a problem, like Slack. That's another potential easier upsell.

You have to think about how do you make the upsell process easy. The other final piece is just understanding the GTM, which is who is your user? What's the buyer? We talked with this earlier. How does user influence the buyer? How do you market to the buyer? Way back at the beginning of this discussion, we talked about in the platform engineering team and how do you influence them. A lot of people think product-led GTM is if you build it, they will come. It's not. You better understand how marketing and sales really play into your discussion. You need to be thinking about – I think marketing is one of the most underrated – product marketing in particular, it's one most underrated skills in an early stage startup.

Most founders don't have a product marketing background, so you need to be thinking early on in your company, how are you bringing that DNA in. These are really important topics to be thinking about and how these speed bumps going to be approached as a part of your company lifecycle.

[1:06:10.1] JM: I want to talk a little bit about the – some of the investments you've made –

[1:06:13.4] VS: Yeah, sure.

[1:06:14.4] JM: - as we begin to wrap up. One of them is Dev.to, which I made a small investment in myself. I'm really happy to see them doing so well. Early on in this business, in Software Engineering Daily, I was thinking a lot about what is the big business that could be built around developer media. I remember talking to Ben in the early days. This was back in 2015 or 2016.

[1:06:48.0] VS: It’s Ben Halpern, right?

[1:06:49.0] JM: Ben Halpern, who founded Dev. We both had pretty similar ideas for it. The issue is for me, I was making five podcasts per week and it was going well. I watched as he built a really good platform. He built a perpetuating platform, because you'd obviously rather be in the platform business than the media business. I mean, in some sense, the platform business is much harder. You look at Medium. Medium is having some trouble. The idea of a self-perpetuating content engine, versus a Software Engineering Daily style treadmill of content, I just remember going back and forth in my head and with my early co-founder of Software Engineering Daily when we were talking about how do you build a software media business.

I think Ben really figured it out with Dev. I'm just really proud of him and glad he let me invest. Describe the market from your point of view that Dev is serving. What does practical Dev do for the developer audience?

[1:08:06.6] VS: Yeah. First of all and I'm glad we're co-investors with Dev. It's great to be working with you on that one. I'm really excited about Dev for a number of reasons. Working with Peter, Ben and Jess, PBJ as they often go by, they're a trio of amazing co-founders. The energy they bring and their dedication to their community is just – it's amazing. Working with them has been an absolute pleasure. It's great to join the board and help them grow the company.

The market that they serve and the opportunity there is something quite unique. This brings back to how – found the company and first started engaging with the team. Before I made the investment, I discovered Dev.to. I'll just go to the website and started looking at it, because I’ve spent some time around the developer community, coming from the software engineering and infrastructure world, there's a lot of platforms out there.

If you look at some of the more places that have been around for a while, you look at Reddit, Hacker News, or what have you, a lot of times there's what I call a certain level of toxicity or a gatekeeping. The church analysis, you put up something and people – People will give an answer of the upload or download answers. There's an expectation based on what you know, where you come from, what computer languages, I'm going to say you speak, or what you're proficient in, what background you have.

In some cases of an antagonistic approach, Dev takes a fundamentally different approach. Dev takes the approach that no matter what background you come from, who you are, where you from. If you call yourself a developer, you are a developer. There's this just incredible positivity and energy that the community and culture brings and level of openness. That's what is fundamentally driving Dev.to and the movement around it.

That movement is also what drives the opportunity I think around the company. Because if you look at it, developers are coming from all kinds of different worlds today. They're not just coming from traditional four-year CS degrees, from traditional universities. They're coming from coding boot camps. They're coming from people switching careers. Software is eating the world and that means people are coming from all kinds of different worlds.

My degree was in material science. I actually wasn't a traditional software engineer. I studied semiconductor device physics. I came from hardware. I learned software in different ways. That's partially why Dev resonated with me, because if there was a place in Dev for people who didn't come from software backgrounds, there's a place for me. It instantly resonated with someone like me. That's also I think what's attracting people to the platform. They had that 5 and a half million users and growing rapidly from there. They're the fastest-growing online platform for developers.

I think that's where the initial opportunity for Dev came, was this ability to build an open platform for developers to express and share ideas in a positive manner and it's working. The retention rates are amazing, engagements are amazing, articles are growing really fast. If you've met Ben and Peter and Jess, their engagement to the community across the platform itself and across Twitter and others is amazing. They're drawing people in.

Recently, they announced a acquisition of code newbie, which basically helps bring in the ability to attract next generation of developers. You can see the synergies there. That mission is extremely strong. There's a second, actually very, very important piece and that's the fact that the actual platform that Dev is built on is an open source platform itself. This is actually extremely key.

Dev.to is an open source platform for building communities. The same people who are both creating articles and engaging with communities within Dev.to, the website and community are the same people who are building a platform for building communities. That in and of itself creates an opportunity for a platform that builds communities. Again, there's long term thoughts around where the opportunity is there.

In general, I think the opportunity for Dev.to is the ability to bring developers together across a number of different spectrums, whether it is no matter the personal background you come from, the educational background you from, the languages, literally the language you speak as a person, or the computer languages that you write in, that opportunity to unify is something that I don't think any other media, or otherwise platform has ever really done before effectively. For us, the opportunity was just too good to ignore.

I'm super excited to be working with the team and to be helping them for that mission. In particular, myself having worked with – working as a part of and working with commercialized open source businesses, I just see a huge opportunity there.

[1:13:01.0] JM: Agreed. Alchemy, which is a company around crypto infrastructure is another investment that you worked on. What is your perspective for the useful applications of cryptocurrency?

[1:13:17.8] VS: Yeah. The background there is – Alchemy is a platform for developers to more easily build decentralized applications. It's less about the cryptocurrency aspects and more about the decentralized aspect. It's about building scalable and reliable decentralized apps in an easy fashion. Right now, building decentralized apps is not easy, if you've ever taken a crack at it. It’s actually rather difficult.

The thing that the founders, Nikil and Joe realized is the market for traditional crypto apps at the time was built around a core users who really understood crypto. You can make parallels back to what I saw in the cloud native world in the early days, building microservices, if you knew what you were doing, was possible. If you did know what you're doing, was really, really freaking hard.

I think what the Alchemy founders, Nikil and Joe realized was there's an opportunity there to really open up the market to more traditional developers on how they can help build decentralized applications in a scalable and reliable fashion in an easy way. For me, the opportunity was less so about the finance aspects.

Again, I'm an IT infrastructure person. I'm not really a finance person. My thoughts have always been around where is the infrastructure world headed? We've seen this transition from mainframe applications to client-server architecture, to virtualized client-server with VMs and vSphere, to semi-distributed applications in the cloud native world. I mean, you look at something like Kubernetes, it's still a master-worker architecture with a semi-decentralized structure.

You see this opportunity for decentralized applications. I think the opportunity is still very early. Don't get me wrong. If you look at it, there are opportunities getting brighter and brighter with the increased focus of the edge, I think in particular. If you look at more and more edge applications, whether it's with 5G, or with industrial automation, that's where I think I see a lot of possibilities on decentralized apps.

Another interesting use case I've actually started seeing is GDPR compliance, which is interesting. You need to have data stored in different places and applications running in different instances. They need to run relatively independently. For that to happen, the applications intelligence needs to be stored independently as well. That doesn't really work in a cloud native world, because cloud native worlds need to be structured still in a centralized fashion. Managers are run centrally.

Actually, interestingly when I was at Docker, we'd get requests for," Hey, can we stretch our clusters across different ages and geographies?" I was like, "Your latency – it's not going to work from a latency perspective." Your shifts in latency was like, Kubernetes, Swarm, etc., aren't really built for that. There are use cases by which decentralized apps become interesting over time. I don't think it's going to fully replace in any way what cloud native is doing, just as cloud native is not going to fully replace what client-server is doing, just like what client-server never fully replaced what mainframe is doing. Contrary to popular belief, there's still a bunch of mainframes out there that are running just fine.

The use cases are there. Knowing that those use cases are there, you're going to need a way for traditional developers to be able to build those apps in a way that they understand easily and you're going to need to build up a tooling for monitoring, for management, etc. That's what Nikil and Joe and the team at Alchemy are doing effectively. That's what we saw the opportunity.

[1:18:02.2] JM: We've got to wrap-up. I've really enjoyed this conversation though. Last question, what has surprised you about becoming an investor?

[1:18:12.8] VS: That's a good question. I think there's a couple of things that have surprised me. I think the biggest thing is you can really differentiate yourself as an investor by how much you're actually willing to help entrepreneurs. This actually surprised me. How earnest you are about being willing to help entrepreneurs. I got into this business, because ultimately I want to help people build things at scale.

I started my career as a materials engineer, something on device physics. Then I moved to software product management in infrastructure. Then I'm a venture investor in infrastructure. My life's mission is to help other people build the things that people use to build bigger things. I didn't get into this business to do deals, so to speak. I got into this business to help other people. I spent a lot of my time on helping people with everything from product strategy, marketing strategy, sales and pricing, hiring what else it is.

To be clear, I don't just help the companies that I work with. I help a lot of the people that I just talk to on the side. That's just what I do, because it's what I like. I think what I've learned is that within the limits of time, you'd be surprised at least from what I've seen is how much that can help differentiate at least, or how much people appreciate that and how much that can differentiate you. If you're earnest about actually providing help and helping people get things done. That surprised me, but it's what I like doing. I think as an investor, I have a philosophy. It's what I call being a servant investor. It's about using a combination of – you're familiar with the concept of servant leadership?

[1:20:01.7] JM: I've heard it.

[1:20:02.4] VS: It's basically where you work for your people. Your people don't work for you. Anyone who's ever been a product manager knows this concept pretty well, because product managers don't have any real power. Product managers don’t manage other people. Product managers only have as much power as they have their ability to influence others. You are the voice of the customer and you bring data to the table and you help make other people more effective by helping shape the product and helping the engineering team and the design team and others in getting the product built. That's what product management is.

While VCs in some cases have official power before the board, I think that a good VC is a good product leader. I think a good VC is someone who brings a combination of their values and experience, but also a sense of humility and the ability to work quietly in the background to help get whatever the entrepreneur needs to help get things done. That's the attitude that I like to bring, what the folks are bringing.

The analogy that I use is I'm a big fan of comic books and sci-fi and fantasy. It's like being the Alfred Pennyworth to founders Batman or Batwoman. That's the attitude that I take is you're not the center of attention. You're the one who's helping them get the work done.

[1:21:18.6] JM: That's the butler that hangs out underground.

[1:21:20.3] VS: That's the butler that hangs out underground.

[1:21:22.2] JM: Okay.

[1:21:22.8] VS: That's the way that I think about it. I guess, what's surprising of being an investor is if you're earnestly willing to help people and bring whatever you have to bring to the table and you actually put those cars down, people do appreciate it.

[1:21:36.5] JM: Vivek, thanks for coming on the show.

[1:21:37.9] VS: Jeff, thank you for having me.

[END OF INTERVIEW]

[1:21:48.5] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications. It's definitely hard to get engineers excited about maintaining those back-office applications.

Companies like DoorDash and Brex and Amazon use Retool to build custom internal tools faster. The idea is that internal tools mostly look the same. They're made out of tables and dropdowns and buttons and text inputs. Retool gives you a drag-and-drop interface, so engineers can build these internal UIs in hours, not days. They can spend more time building features that customers will see.

Retool connects to any database and API. For example, if you want to pull in data from PostgreSQL, you just write a SQL query. You drag a table onto the canvas. If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily. You can even host Retool on-premise if you want to keep it ultra-secure.

I've heard a lot of good things about Retool from engineers who I respect. Check it out at retool.com/sedaily.

[END]