EPISODE 792

[INTRODUCTION]

[00:00:00] JM: Red Hat was the first commercial open source software company. For years, investors and entrepreneurs assumed that there would never be another Red Hat. Red Hat's business was built around enterprise operating system distribution and support. Since the operating system is at the core of how users within the company are doing their job, Red Hat had a lot of leverage and a strong business model. But how many enterprise software products could be so critical to a business that they could manage to offer their software as an open source option yet still make money? This was the tension that people assumed would occur within open source businesses. If you offer it for free, why is anybody going to pay you for anything? As it turns out, there are many ways to make money in open source.

MySQL aid away at the dominance of Oracle's database business for similar reasons to Red Hat's success. Much like an operating system, the database layer is critical infrastructure. There is also Cloudera and Hortonworks. They were able to monetize the open source Hadoop project, because Hadoop was kind of difficult to deploy and manage.

As cloud infrastructure matured, it became easier to start companies that offered open source software as a service. Elastic offers an easy way to use elastic search. Redis Labs offers Redis as a service. MongoDB, the company, offers MongoDB as a service. As it turns out, engineers love to see the source code for databases, but they do not enjoy deploying and managing those databases, and people are very happy to pay cloud providers to save them time. Yet, still, there is continued skepticism of open source businesses.

Today's debates center around whether individual providers, like Elastic, can offer a service that competes with an Elasticsearch service offered by AWS. The reality is we could just as easily be asking the inverse question: How can AWS compete with an entire company that is dedicated to the deeply technical problem of solving search?

Most of these open source product categories have an enormous total addressable market, an extremely good unit economics. This applies to both cloud providers and point solution providers. Investors often talk about how much they love subscription businesses.

When a company starts purchasing infrastructure as a service from you, it's like they're buying a subscription where the annuity increases over time. People buy more and more of a cloud provider solution every year because they're storing more and more data in it. When an investor says that they are worried about a giant cloud provider offering the same service as an open source company, it's similar to the investor being worried that a new sales tool is going to be duplicated by Salesforce. The market always needs new sales tools and the market needs those tools to be offered both by Salesforce and by smaller individual CRM companies.

In the world of commercial open source, there's plenty of room for both point solution providers and cloud providers, but they are competing for the same customers, and the competitive battlefield is expanding to the nuanced world of software licensing. By changing their licenses, open source projects, like Kafka, MongoDB and Redis, can prohibit AWS from certain usage patterns. This might offer some protection for companies based around the point solutions, companies like Confluent and Redis Labs.

Beyond the fracas of the battle between cloud providers and point solutions, there are newer open source companies with models that do not fit tightly within any historical business models. HashiCorp makes a suite of differentiated open source tools that have not been seriously contested or offered as a service by cloud providers. GitLab makes an open source platform that is built with monitoring, logging, CI and code hosting out-of-the-box. As the world of open source business models expands, more companies will find opportunity in open sourcing the code that runs their products.

In many cases, companies will find that it strengthens their advantage to open source their code rather than weakening it. The defensibility of many businesses relies more on data and network effects than the contents of their code base. We may see the default question gradually shift from, “Why should I open-source my code base?” to “Why shouldn't I open-source my code base?”

Mike Volpi is a partner at Index Ventures and has invested in many open source businesses over the last decade. He's on the board of Confluent, Cockroach Labs, Kong and Elastic. Mike joins the show to share his perspective on open-source business models of the past, present and future.

Before we get to that conversation, I want to mention a few events that we're having in the near future. One is a live event with Haseeb Qureshi at Cloudflare. I'll be sitting down for a conversation with Haseeb Qureshi. He an investor, and a software engineer, and a popular guest on the show from past episodes. That event is April 3rd.

We're also having an in-person hackathon on April 6th, the FindCollabs hackathon. FindCollabs is a new product that I have been working on. It's a company I'm starting, and it's a place to build projects and collaborate with people. If you have an open source project, or a business, or a song, or an artistic project, FindCollabs is a great place to build your project and find collaborators. Also, the hackathon can be accessed virtually. It's a virtual hackathon, as well as an in-person hackathon, and the prizes include $4,000 for first place, and $1,000 for second place. There're other prizes as well. So you can go to softwareengineeringdaily.com/hackathon to find more about that. That's April 6th. It will be at App Academy as well as virtually.

With that, let's get on with this episode.

[SPONSOR MESSAGE]

[00:06:39] JM: Clubhouse is a project management platform built for software development. Clubhouse has a beautiful intuitive interface that's made for simple workflows or complex projects. Whether you are a software engineer, or a project manager, or the CEO, Clubhouse lets you collaborate with the rest of the product team. And if you're an engineering-heavy organization, you will love the integrations with GitHub, and Slack, and Sentry and the other tools that you're feeding into your issue tracking and project management.

To try out Clubhouse free for two months, go to clubhouse.io/sedaily, and it's easy for people on any team to focus in on their work on a specific task or a project in Clubhouse while also being able to zoom out and see how that work is contributing towards the bigger picture. This encourages cross-functional collaboration. That's why companies like Elastic, and Splice, and Nubank all use Clubhouse. Those companies have all been on Software Engineering Daily, and we know they are competent engineering organizations. Stay focused on your project and stay in touch with your team. Try out Clubhouse by going to clubhouse.io/sedaily and get two months free.

Thank you, Clubhouse, for being a new sponsor of Software Engineering Daily.

[INTERVIEW]

[00:08:10] JM: Mike Volpi, you are a general partner at Index Ventures, welcome to Software Engineering Daily .

[00:08:14] MV: Thanks for having me, Jeff.

[00:08:16] JM: Red Hat was the first commercial open source company. Why was Red Hat's business model unique at the time?

[00:08:23] MV: Well, as you know, the open source software, particularly the variance that were available in the 90s, were free. Anybody could take the source code and download it and use the product. So it was commonly perceived that there was no business around open source software. That how can you create a business around something that's free?

And then what happened is that, as in their case, as Linux became more and more broadly deployed as server software and particularly being deployed in large enterprise contexts, a lot of enterprise customers wanted to have some level of support around that. So what if you run into issues? How about configuration? Scalability? Manageability? Etc., all the various abilities. Essentially, Red Hat built a business model around just support. So the software were still fundamentally all free and downloadable. However, they built a business around support. Historically, that wasn't thought to be the best business model possible, much easier and better margin to sell software. But they built a very good business. As you know, the company subsequently went on to be worth over 30 billion.
So it actually became a very interesting business.

[00:09:35] JM: Why did people think for so long that the Red Hat model could not be replicated?

[00:09:41] MV: The thesis being that in order to – If you have a software that's free and you charge for support, you can charge only a fraction of what a proprietary piece of software would for equivalent functionality. Therefore, you have to have a really big, big market – What market is bigger than operating systems at the time? In order to monetize sufficiently to build a big business.

So if you think about the big categories of infrastructural software, operating system, databases and then you start getting into more and more niche areas, and it was generally thought that Red Hat could do what it could, because operating systems, everybody needs one. So you can build a support business model around it, or other forms of software that occupy smaller niches in terms of overall usage or market size. It was thought that you couldn't really build a business model that sufficiently monetized.

[00:10:37] JM: You are the chief strategy officer of Cisco in the 90s and early 2000s. What was your perception of open source back then?

[00:10:46] MV: We used open-source at the time at Cisco. We didn't pay for it. I would say we probably had the common perception, which is that it was an interesting phenomenon, but that they weren't really businesses around it, that it was just software that you can use for your products, in some cases embedding the software. But it wasn't thought of really as a business model in the 90s.

[00:11:07] JM: In 2009 you became a general partner at Index. What was your thesis on open source when you joined the firm?

[00:11:13] MV: A lot changed in the decade starting with the 2000's around open source. The firm, Index, prior to my arrival, was an early investor in MySQL. MySQL was also built on a very similar business model to Red Hat. It was the free open source databases and the business support or the business monetized on support subscriptions.

The database market like the operating system market was large. So MySQL built a good business around it. It was eventually acquired by Sun, and then subsequently Sun was acquired by Oracle. It was a very good outcome for Index. So we had a sense that this open source thing had legs.

What started to kind of emerge was a series of next-generation databases, let's call them. Hadoop, which is a database with a computing model built on top of it, MongoDB, Cassandra and so forth, all of which had sort of turned the crank a little bit on the open source business model and said, “The bulk of the software is free, but there's a layer of commercial or differently licensed software that sits around the core that allows the company to monetize in a new way, which is charge for some proprietary software.

In the early days, and if you look at the players that played that game; Cloudera, Mongo, Cassandra, all had sort of an orchestration layer, a management layer, some security features, alerting notifications. So, essentially, broadly, a set a features that only would be used in enterprise production that were held back in proprietary or commercial mode. The core pieces of the software were still very much open source. That was believed would monetize more effectively than just pure support subscriptions, and it turned out to be the case.

So in ’09, we sort of looked at this market. I think in 2011 we made an investment in Hortonworks, which was one of the two large Hadoop distros. Subsequently, we made a number of other investments, like Elastic, and Confluent and so forth all under the same thesis of the evolution of open source going from all open with support subscriptions to mostly open with a layer of proprietary software on top of it plus subscriptions.

[00:13:36] JM: The Hadoop vendor wars included Cloudera, and Hortonworks, and MapR, and a number of other players. Describe the Hadoop vendor wars as you remember it.

[00:13:47] MV: So Hadoop obviously was one of those technologies that found its origins in open source and became very, very popular in the era. If you actually kind of look at the lineage of Hadoop, it started as a project within Yahoo. A group of people left Yahoo and founded Cloudera, and a group of people stayed at Yahoo.

The second group, the ones that stayed, eventually spun out with Yahoo's consent into Hortonworks. Then at the same time, another company called MapR kind of came up. They had three very different strategies. Cloudera had true open source Hadoop with a layer of commercial software around it. Hortonworks, which believed that it had to catch up and surpass Cloudera thought that the best way to do that was to be all open, so to kind of revert back to the Red Hat business model, if you will. MapR took it evenly further and had more commercial software than Cloudera did.

Of the three, the Cloudera, which was the first one out of the gates and the first one to commercialize, and Hortonworks took off. Both went public. Eventually, those two companies merged in part because at the core was the same open source piece of software. MapR kept going on its own trajectory. We don't know a lot about it, because that's a private company, but presumably they haven't had as much success as Cloudiera and Hortonworks.

But the fact that that main open source project of Hadoop got kind of split into two companies, both of whom had kind of moral authority, if you will, over the project, Cloudera and Hortonworks, probably hurt the market in the end, because what the two did is that they pulled in different directions in terms of the evolution of Hadoop and they competed with each other very, very aggressively. Therefore, the customers in some sense may have benefited from it, but the vendors themselves struggled to monetize as effectively because of that kind of split brain. Obviously, that's the logic behind the merger of the two companies that they reunified kind of the brain of Hadoop and its evolutionary path so that they can build a more robust business model and hopefully a more coherent and concrete product strategy around one vendor as supposed to two. So that's kind of what's happened in that universe.

[00:16:09] JM: How did the economics of the Hadoop vendors compare to those of Red Hat?

[00:16:14] MV: That's a good question. It's quite a different – In principle, they're both open source companies and so forth, but they're fairly different, because Hadoop is kind of a cluster-based system where you have to run a bunch of different components that sort of sit underneath Hadoop to run it together. The clusters get fairly large, tens of thousands of nodes in some cases. It's not kind of an operating system thing like Red Hat, and Red Hat obviously also has evolved. They've got a bunch of other sort of cloud orchestration things in the business. So they're reasonably distinct, I would say, where Hadoop – The Cloudera now, the combined company, really just focuses mostly on Hadoop and its evolution, whereas Red Hat has become more of a holding company with a lot of different piece parts into it. So still pretty different businesses I would say.

[00:17:07] JM: I think that extrapolates to the other open-source businesses as well, because if you look at Confluent, or MongoDB, or Elastic, you can't really categorize the economics in terms of their open source nature. Open source is merely a characteristic of the business. Is there any correlation that we can draw between the fact that a business is open-source and the economic characteristics of that business?

[00:17:34] MV: Yeah. I mean, let me answer the question this way. We as an investor learned a lot of lessons from our journey with Hortonworks, and we realized that to build robust open-source business, you had to have a few characteristics of that business that were true. One was that the company had to be really the sole moral authority over that open source project.

I think the industry lost a lot of steam because of the split brain thing that happened in Hadoop. So we tried to avoid that situation. Generally, what that means is that the person or the core people that are the true sort of authors of that open source project have to be employees of the company. So if you look at Mongo, that's the case, Elastic, Confluent, Hashi, Kong. In each one of those cases, the principal authors are very much employees, and in many cases leaders in that company. So that's an important precondition.

The second thing is that you have to have a piece of software. You'd like to have piece of software which can gain a lot of organic developer adaption, and what I mean by that is literally the software becomes well-known because of some positive attributes that it has, and developers go grab it and download it and then use it. That creates kind of a groundswell of positive momentum towards the company.

So in all the cases, whether it's Kafka, Elastic, Mongo, Hashi and so forth, you just see this enormous amount of downloaded and used instances of the software from the developer community. Hadoop was a little harder to do that, because you need at least 50 to 100 nodes of

Hadoop cluster to actually run when you can't run on your laptop, which is what a lot of developers do.

So we'd look for situations where just a handful of developers can grab it, literally, on their laptop, try it out and fall in love with it. That's another very important theme. So these companies, like Elastic, and Confluent, we sort of view as the third generation. They turned the crank one more time from where Hadoop was into a very fluid and dynamic market. Obviously, they've also taken a page out of what Cloudera did, and most of these companies, in fact, arguably all of them, have this layer of commercial software on top of it, which makes it easier to monetize.

[00:19:59] JM: The first factor you've mentioned, the idea that a company would be the particular moral authority over a project, that becomes harder to think about when you consider Kubernetes, because Kubernetes is effectively controlled or overseen by the CNCF. CNCF was heavily shaped by Google in its founding, but Google is not exactly the controller of the CNCF. Of course, Kubernetes is perhaps a more transformational project than any of these other ones that we've even discussed. I don't know if you would agree with that, but how does Kubernetes as a technology fit into your framework of open source investing?

[00:20:41] MV: Well, not every open source project has to be a company, and there can be very successful open source projects that are not companies. They're just open source projects. I put Kubernetes in that category. I think Kubernetes, you can't argue the fact that it is become the dominant orchestration architecture and product for containers. It's largely what the world is using. There have been a few companies that have sort of attempted to jump on to that bandwagon. Heptio, for example, recently got acquired principally for that reason, but none of them have demonstrated that they actually have the moral authority over Kubernetes for exactly the reasons that you mention, which is Kubernetes is largely the moral authority of Kubernetes rests with Google.

I think Google has done a very effective job at building community around Kubernetes through CNCF and other methodologies, but is not clear to me that there is going to be ever a billions of dollars scale business around Kubernetes, or perhaps Google itself as that. That doesn't mean that it's not going to be successful. It clearly is, and i think it's here to stay. So great open source projects don't always translate into great businesses.

[00:21:52] JM: Confluent is the leading Kafka vendor. Do you think of Confluent as a database company?

[00:21:58] MV: Confluent is a very interesting multifaceted company, because in its heart and its origin, Kafka, which is the software that it purveys, was effectively a message bus for the enterprise at large-scale, sort of streaming scale data. But because some of those applications that were interconnected via Kafka required essentially asynchronous transfer of data. So there is a write that happens, and quite a bit of time later, there is a read that needs to happen.

Kafka needed the ability to store that information for some amount of time. That amount of time could be minutes, could be days, could be years. When you store stuff, you need a database. So underlying it all is a key value store, a data store, which is very important, because once you store data, you can start to build applications around it and you can begin to manipulate that data. So in some sense, Kafka serves and Confluent serves three purposes. One is the movement of data. Two is the temporal storage of that data, and three is the processing or manipulation of that data. You could say Confluent is a company that does all three. It moves information, it stores information and it allows for the compute of that information.

In fact, one of the fastest growing areas for Confluent is this product called the KSQL, which they introduced, which is essentially a SQL error that resides over this information that store it inside of Kafka, and that's gained popularity very quickly. When you start to think through why, it becomes pretty obvious that the place where data moves is an interesting place to make transformations around that data.

So I think Confluent is one of the more exciting companies that I've seen, because it's sort of – Depending on which way you look at it, it's a different kind of a company all three of which represent something very interesting.

[SPONSOR MESSAGE]

[00:23:59] JM: Your audience is most likely global. Your customers are everywhere. They're in different countries speaking different languages. For your product or service to reach these new markets, you'll need a reliable solution to localize your digital content quickly. Transifex is a SaaS based localization and translation platform that easily integrates with your Agile development process.

Your software, your websites, your games, apps, video subtitles and more can all be translated with Transifex. You can use Transifex with in-house translation teams, language service providers. You can even crowd source your translations. If you're a developer who is ready to reach a global audience, check out Transifex. You can visit transifex.com/sedaily and sign up for a free 15-day trial.

With Transifex, source content and translations are automatically synced to a global content repository that's accessible at any time. Translators work on live content within the development cycle, eliminating the need for freezes or batched translations. Whether you are translating a website, a game, a mobile app or even video subtitles, Transifex gives developers the powerful tools needed to manage the software localization process.

Sign up for a free 15-day trial and support Software Engineering Daily by going to transifex.com/sedaily. That's transifex.com/sedaily.

[INTERVIEW CONTINUED]

[00:25:49] JM: In modern data infrastructure platforms, we think about the data lake, the data warehouse, we have databases, we have materialized views, we have message buses, like Kafka, and when I talk to the Databricks people, they present a future where this is simplified. These different areas, like data lakes, or perhaps message buses, get consolidated into pieces that are simpler to reason about. What's your vision for how the data platform or I guess the prototypical data platform looks in 5 or 10 years?

[00:26:29] MV: Yeah. I would respectfully disagree with the folks from Databricks, because if you actually think about what is a place – What are these locusts that move data or restore data, they're not the dog, they're the tale. The dog is the application. There's some application that needs to make use of this data, and depending on the attributes of the application, the requirements, the needs, the data needs to be stored and moved in a very different way.

So the notion that you have one single way in which – Universal way in which you should store the information, or one universal way in which you should move the information just isn't accurate, because we are specializing – Data is kind of the lifeblood of businesses these days. Companies differentiate themselves on how their application use data. So anything that makes them compromise one particular usage of the data, because the architecture is set up a certain way, actually compromises their business.

The thesis that it's all one is sort of like the minivan thesis, right? The minivan works for everybody. Put people in it, you can drive it around, it has cup holders and all that, but if you're in the business of doing a Formula One race, you'll lose that race with the minivan. If you're in the business of making a fuel-efficient car, you'll lose that business with the minivan. If you want to go off-road, you'll lose that business with the minivan.

What's happening is that you're actually seeing a Balkanization. You're seeing a larger number of data storage environments, or data management environments, or data movement environments emerge, not a lesser number, and it's normal. I think that as businesses specialize and they use data more intelligently, they are going to architect multiple different types of data infrastructures in their business in order to best facilitate the competitive advantage for them. You're going to see more variety, not less. So then interconnecting between them does get important, but you're not going to consolidate it all into one data lake or one data architecture in the near term future, I don't think.

[00:28:34] JM: Describe the competitive dynamics between the major cloud providers and the open source companies.

[00:28:41] MV: Yeah, I mean there's an important nuance here of separating out a little bit of the view of how the cloud providers played, I guess, with open source amongst themselves. There's Amazon, and what they do, and then there is sort of Google and Azure and see what they do. So let's take those three, which basically that's – In our world, at least those are the ones that have the most traction today.

Amazon, broadly speaking, has taken the approach of essentially taking the open source software that's out there and launching their own service with it without really any material partnership the vendor of the open source project. So if you look at Elastic, Amazon has Elasticsearch as a service. That service has nothing to do with Elastic, the company, but it uses the same open source software. It's a few versions behind, and they deploy it. Elasticsearch has no influence or monetization on that product.

We saw the recent kerfuffle with Mongo around that idea. EMR, Elastic Map Reduce from Amazon is a variant of Hadoop, which they've introduced. Again, no relationship with the vendors in particular. So I think Amazon's philosophy is this stuff is free. We take it, we put a service on top of it, we provide no monetization back to the open source vendors around that and we're just good enough to manage it ourselves.

The other two, Ggoogle and Azure, have taken a little bit of a different tack to it and both have either explicitly or implicitly stated that they're partnering with the open source vendors to provide the sort of legitimized version of that service. So you're seeing them begin to launch services based around Mongo, Elastic, Kafka, Hashi, etc., etc., all of which actually carry the full version of that software. So both the proprietary commercial, and the open source version offered as a service by those.

In that context, if you're in open source vendor, Azure and Google look like partners, look like distribution and so forth. So I would say that Azure and Google have taken a very open source friendly approach and open source company friendly approach, where Amazon has definitely taken advantage of the open source movement, but not been particularly friendly to open source vendors.

What's that kind of created is I would say some redefinition of the license architectures that some of the open source vendors have created in order to address this issue of, "Look, if you're going to take this piece of work that we'd put all this effort into and monetize it, we should get a cut of that," and that's what the sort of these new relicensing stories are all about.

[00:31:19] JM: And is licensing a viable strategy?

[00:31:22] MV: Yeah, I think so. I mean, essentially, if you at kind of the original – The original open source license that a lot of people use was Apache, because that was the one that sort of gave everybody credibility. These days, most companies are not starting with Apache as much anymore, because the Apache licensing architecture leaves very little room for the companies, the open source companies, to defend themselves against an Amazon-like approach.

So the newer generations of the licenses essentially have a carve out, that if you're going to launch a commercial service, that is a commercially offered cloud service, using that software, you have to obtain a license from the company with it. I think that's a pretty viable way of sort of segmenting, if you will, that flow, so that if Amazon wants to do it, they have to write their own software, like they've done actually with DocumentDB, and they can make it API compatible with MongoDB, but it's not going to be the same software.

So, essentially, that's kind of naturally happening. What we are seeing in the younger generation of organizations of open source companies, they're not using Apache anymore. They're using more of these licenses that essentially allow them to basically tailor the license to the type of user that's on the other end rather than sort of blanket open source that existed before.

[00:32:38] JM: Let's say you're Satya Nadella of Microsoft. You just acquired GitHub. What's your plan for the next five years to maximize the value of that acquisition?

[00:32:48] MV: Well, I mean, the biggest and most obvious thing is that keeps the developers very close to Microsoft, right? If you kind of think about, most of us business people think of our LinkedIn as our home where we kind of exist in the world. For software developers, it's GitHub. Microsoft I think has, over the years, always understood the criticality of the developer. Now, they've become effectively the home of that developer.

So I believe that the Microsoft acquisition was a very strategic one. I mean, the price they've paid was obviously indicated that, and my guess is that they're going to use that to, A, obviously enhance the enterprise versions of GitHub and sell more of that so that all the private repose and so forth that are using the enterprise will be sort of Microsoft-ized and monetized effectively and at the same time continue to scale the evolution of GitHub.

So I think supersmart acquisition on their part. Especially for Microsoft that's not known for owning cutting-edge developers. They're still kind of the .net crowd. I wouldn't say – I wouldn't call it hipster compliant, but they now sort of have inherited this platform where a lot of the cutting-edge and sort of web style developers live. So very, very strategic for them.

[00:34:11] JM: GitLab is making an effort to capture the full value stream of software development within the GetLab platform, rather than building out lots of integrations. Do you think the GitLab strategy is a bold and smart strategy, or do you think it's overly risky?

[00:34:28] MV: It's an interesting strategy. Obviously, they're having some good success with it. We'll see how it plays out. I think it's going to be awfully hard for them to kind of disrupt that critical mass that GitHub has. The argument might be, "Well, Microsoft is going to screw it up." I don't think Microsoft is going to screw it up. I think that they understand very well the value of what they bought, and I think that they're still going to be the go-to place. I do think that, again, a more heterogeneous environment is always a better bet that a more-narrow constrained environment. So I like the Microsoft play better, but who knows? We'll see how it plays out.

[00:35:06] JM: Were you surprised when GitLab painted that vision of this fully integrated we're going to solve monitoring and analytics and continues integration and everything all in the GitLab platform?

[00:35:19] MV: I was surprised only to the extent that it's pretty ambitious given that everybody has their favorite tool for one of those, and the notion that you're going to compete on so many fronts feels overextended. But we'll see how it plays out.

[00:35:31] JM: Yeah, it could work. Let's say you're Ginni Rometty at IBM. You just acquired Red Hat. What's your five-year plan for making that acquisition worth it?

[00:35:39] MV: Look. I mean, I think IBM wanted to get back into the modern software game that sort of lost the thread there a little bit. Initially, they made a big push around Hadoop. If you remember, in the early days at least, they had some I think intent to maybe commandeer the Hadoop/Spark ecosystem. I don't know that that went very far for them. I think that they're trying to find a way back into the game via this Red Hat acquisition. They've paid a pretty substantial premium. You could understand why the Red Hat people took the offer.

I don't exactly know where that goes. Presumably, they will continue to acquire more open source components and attempt to be the home of open source. It isn't clear that IBM customers are the earliest adapters of these kinds of technologies. Certainly, Red Hat had become a very mature company at this stage. So maybe they'll keep looking more of these. Hard to say, but they do kind of live – I'd almost characterize it as a slightly different ecosystem than the starup universe that we live in.

There are big companies that use a lot of IBM services. They will continue to do that, but they aren't the early adapters. So for the business that we live in among startups and so forth, it's not particularly concerning, and maybe they'll provide an exit path for some of our portfolio companies later on.

[00:36:58] JM: Microsoft has been able to improve its brand through engagement with open source and development of the cloud. Can Oracle do the same?

[00:37:06] MV: I mean, I think they could do it. I don't know if they want to do it. It's interesting to characterize it this way; Oracle, through the acquisition of Sun inherited MySQL, and MySQL in the relational database world was definitely the winner for a long period of time. Subsequent to that acquisition, and it's been a few years now, you've seen sort of the reemergence and the relational database world of PostgreS, and you kind of have to attribute like why. PostgreS, which had essentially kind of lost the battle to MySQL at one point is now – As much of the default choice as MySQL is. Arguably, it's more the default choice and why. You could argue that maybe it's technically better in one way or the other, but I think that the developers look at MySQL with some degree of skepticism, because it's owned by Oracle at this point. I don't know that Oracle is particularly endeared by the average developer these days. In fact, if you walk around SOMA these days, I don't think you're going to run into a lot of developers saying, "I'm starting out from an Oracle database."

So, they could if they wanted to. I don't know that they necessarily are going that way. Arguably, a lot of these open source projects do present kind of an innovator's dilemma to Oracle, because Oracle software, there's a lot of server software that is monetized in a proprietary way, and if they were to embrace open source wholeheartedly, they could potentially severely damage their cash cows. So I think it's a bit challenging for them, frankly, from a business model, to embrace open source the way Microsoft had. Because if you think about Microsoft, yes, they did have some server software that open source could potentially take away the cash flow stream from, but largely, Microsoft cash cow are operating system and productivity software. So there, they're much less threatened by the open source ecosystem than Oracle is.

[00:38:59] JM: What's the best way to invest in the service mesh category?

[00:39:03] MV: Let me take a step back and talk a little bit about service mesh. So service mesh is very interesting, and that it is a new way to kind of interconnect containers. There isn't an obvious leader in that right now, and that there are a couple of service mesh projects. There's Envoy and Linkerd. Linkerd has become a company. It's called Buoyant. Envoy doesn't really have a company behind it yet. The author still works at Lyft. Then Google has launched Istio, which is sort of the control plane above it.

Overall, the adoption is still super early and sort of part of it is probably because there isn't company really behind it. But it's sort of in the let's see category. It's definitely an interesting architecture and it works well for containers. It doesn't work nearly as well for the other kind of aspect of containerization the happens, which is sort of the Balkanization of the monolith, right? A lot of applications were built as a single monolith and you want to break them up, but you don't break them up with a thousand containers. You break them up with four or five. For that architecture, a more classical API microservices architecture might work better.

We have a portfolio company called Kong, which is kind of the leader in that microservices gateway business, which has now also kind of embraced the service mesh. Obviously, we're biased here, but we feel like they have a fantastic opportunity to capture a lot of that service mesh momentum, and we'll see how it plays out. But they've certainly garnered a lot of – That they're clearly the winning player in the microservices area. As service mesh increases in adaption, we hope that people will continue to use the Kong's kind of service control plane and will adapt either a more centralized microservices architecture versus a distributed one, like service mesh, depending on what the application needs.

So the way I sort of see it is the high-level theme is the migration to microservices and containers. The low-level question is which parts of my network do I go centralize? Which ones do I go mesh with? You're probably going to use a combination of the two depending on what the application requires.

[00:41:11] JM: Now if I recall, the Kong service mesh model uses NGINX, right? So I've talked to other people who say, "Oh no, it's actually really, really important that you use Envoy nodes for your service mesh. Does it actually matter for most customers whether it's NGINX or Envoy?

[00:41:29] MV: From a Kong perspective, at the end of the day I don't think it matters too much. It's the transport layer essentially of communication between microservices, and I don't know that it's – Again, for some applications, mesh is better, and I would say probably for applications that are natively built from a containerized perspective, the mesh might be a better approach.

For applications that are transformed into microservices, in other words, they weren't born as a microservice, the more traditional reverse proxy architecture, I.e., the NGINX architecture, might be better. In Kong's perspective, it's whatever you want. You're using our control plane above it, because essentially the key thing about a microservices' Gateway is that that's where you implement level of control. So your policy is throttling, your billing, your diagnostics and analytics, all that stuff lives there. In some sense, you want a consistent services layer above whatever transport architecture makes sense. Today, most of the market is NGINX and very little of the market is Envoy. At some point in the, future that my flip or they might be balanced, but I think from a customer's perspective, the importance is can I set my policies, my security and all of my systems architecture commonly between whatever the transport layer is and go with that. So I don't know that it's a winner takes all honestly on the transport layer. I think you're going to probably see some diversity depending on what the application architectures look like.

[00:42:58] JM: What's the connection between service mesh and zero trust networking?

[00:43:02] MV: That's a good one. I mean, I would imagine that in a zero trust environment, the various elements of the mesh would have to have some degree of security built on top of it, because you essentially don't trust all of your neighbors. Whereas in a centralized architecture, the centralized node behaves as the arbiter of trust, and so the kind of the more modern architectures would probably lend themselves more towards a mesh of a zero trust.

Again, I go back to the same notion, which is that you kind of want a common trust policy, and then you want that implemented on whatever is the most appropriate transport mechanism rather than having a different trust-based architecture for every different type of transport layer.

By the way, Envoy currently is winning, but we may – It's early days, right? So it's not like Linkerd is gone either. To my view, the more Balkanization there is at the transport layer, the more important it is to have a common control plane that sits above it.

[00:44:06] JM: Can you explain that in more detail?

[00:44:08] MV: Yeah. So if you have, let's say, an NGINX style centralized reverse proxy approach, or Linkerd, or Envoy, and in every case you have to define a different policy for those transport mechanisms. When I say it's policy, it's was my security stance? What diagnostics do I get? How do I implement billing? How do I throttling or pushback? How do I ensure the health status of the transport layer? And each one of those you have to implement separately, and I have three of those in my environment. That's a pain, because I have to go define the control plane for each one of those.

But if I can have a single control plane, which I say, "Here is where my security stance is. I want a single place where I can see my monitoring," this is my throttling approach, this is my security parameters and not worry about which transport layer sits underneath it, that's a benefit to the customer, because you define things once. Then if people add to that network or change or maybe this part of the company wants to go with mesh versus these ones, the ones to go centralized, you don't care, because at some point you have a common control plane.

[00:45:10] JM: Have the existence of cryptocurrencies changed any of your beliefs on open source business models?

[00:45:15] MV: No, not really. I mean, I think crypto – The way I sort of looked at – Let's parse a little bit cryptocurrencies versus a blockchain, right? So cryptocurrencies are currencies and they're used for transactions of various types. They're based on a blockchain, which is open source. But if you really look at a blockchain, at least I think of it as a distributed database that has encryption characteristics to it that uses sort of a consensus decision making modality. It's an interesting database. It works well for certain things. It doesn't work well for others. I don't think it's going to take over the world. I do think it has some very relevant applications. The most relevant application are cryptocurrencies themselves. So it works very well for cryptocurrency.

So, for example, blockchain works really well for Bitcoin. Does it work really well for a file sharing? I don't think so. Does it work really well for networking? I don't think so. So I view it as an interesting database or database architecture. I think it'll have some good applications, but it doesn't at all, in my view, change the business model around open source, nor do I think it's dominant in any way open source project. It's one of many that's out there.

[00:46:29] JM: You don't think censorship resistance is a property of blockchain or a cryptocurrency applications?

[00:46:34] MV: I think the average user doesn't care about censorship that much. I think that societal problem that arises at a top-level, but, I mean, even when you look at some social phenomena, like Facebook, Instagram, etc., etc., in theory, you should be worried about censorship or privacy and so forth. But if you look at the usage characteristics of those things, it's still on fire, which means that the average person doesn't care. Maybe they should care, that's a different argument. But today, they effectively don't care.

I think that if you want to introduce some degree of encryption, you don't need necessarily distributed consensus-based architecture, right? The distributed consensus-based architecture basically says, "I don't trust any governing body." Broadly speaking, this becomes a philosophical issue, but I live in the United States, and I like living here, because by and large while my government occasionally screws up, I trust it. That's not true of every country in the world, but it is here, and therefore I don't know that you need necessarily in every case to have a kind of a zero trust environment that is facilitated by a blockchain-like architecture. In fact, in truth, having such a distributed architecture compromises speed and performance. A centralized architecture is faster, because you don't have to take time to go through all that consensus building.

[SPONSOR MESSAGE]

[00:48:03] JM: The blockchain is a new computer science primitive. It allows us to build applications that we could not have built before, and we're in the early days of blockchain applications. It's a great time to get started. Blockstack is an open computing protocol for building applications where users truly own their data. They own their identity and even their content and connections. With a Blockstack ID, users can have a more transparent identity system rather than the modern internet identity systems that are closely tied to advertising.

At blockstack.org/sedaily, you can learn how to build decentralized applications easily. Blockstack is open source, it's free, and it's an application stack that won't serve you ads or demonetize online media personalities, or be subject to the whims of an individual CEO.

Developers who build on Blockstack can even get paid to build better applications using Blockstack via the app mining program. To find out about Blockstack including these programs, you can go to blockstack.org/sedaily. You can learn how to build decentralized applications that are private and secure and easy to build, thanks to Blockstack.

Cryptocurrency are a huge unexplored space. If you're a developer, there's no better time to get started. Just so you know, it's not easy to build decentralized applications today much like it was not easy to build internet applications in 1994, but we know that things get easier overtime and Blockstack is one of the easier ways to develop on the decentralized internet today.

So if you're getting started, it's a great place to go. Go to blockstack.org/sedaily and learn more about how to build decentralized applications.

[INTERVIEW CONTINUED]

[00:50:10] JM: That said – And by the way, I think your view is directionally consistent with a lot of the enterprise distributed systems developers. Like if I go to KubeCon and I ask people about cryptocurrencies, I'll get laughed out of the room, basically. If I'm in a talk about service mesh or some subtlety of the Kubernetes, people don't care about cryptocurrencies. Yet I find it surprising, because maybe they're not totally useful. But cryptocurrencies at least seem like an interesting distributed systems breakthrough.

Does it surprise you that there is not that much overlap between the Kubernetes community and the cryptocurrency community even just as a scientific endeavor?

[00:50:54] MV: I don't know if it surprises me or not. I mean, at the end of the day, the question I try to ask as an investor is what does this particular product architecture technology enable that could not be done before, right? What could you not do before, because this thing has now emerged?

When I look at blockchain, I understand why you can build a cryptocurrency around it, particularly in an environment where you don't trust other people, right? Because, essentially, it's a system that works well without a governing authority, where it's essentially designed to deny a governing authority to do anything about it.

If you're a corporation for your own infrastructure, you are the governing authority. You want to be the governing authority, because it's your company. So in a sense, in an enterprise environment, it works to counter purpose, like you actually want centralization and visibility and control over the systems. You don't want a completely distributed architecture that doesn't trust anybody.

So I think it has an application, but it isn't clear to me that in the enterprise environment, you can certainly do things because of the blockchain that you couldn't do before. You can totally do almost everything that you could do before without it. So it isn't clear to me that there's an application for it.

[00:52:12] JM: But even just from a community perspective. So, I mean, you've you seen the Linux community, you've seen the Hadoop community, you've seen the Kubernetes community. Many of these people are driven by curiosity and a communal sense of talking about distributed systems and building cool stuff. It just surprises me that you have – The cryptocurrency community is like its own open source community where they're very enthusiastic about cryptocurrency related. If you look at it in a microcosm, it looks a lot like the Kubernetes ecosystem or a Hadoop ecosystem, because you have these people who are just frenetically excited about this thing. Yet the cryptocurrency community seems like this group of open source wallflowers that just don't really cross-pollinate with the other crowd. But anyway –

[00:52:58] MV: Look, it may be the case. At the end of the day though I would say that community building, at least in the open source world that I'm more familiar with, has two characteristics. One is that you have very advanced people who are who contribute change evolve the nature of that open-source project. The other one are people that are very simply saying, "Wow! This is so convenient. I'm going to grab a MongoDB and make me something tomorrow morning."

Same with Elastic, like if you actually went – Like when we were doing way back in 2012, our due diligence on Elastic, you certainly had people who said like, "This is amazing, and it involved this way. We're going to change it this way." But a large portion of the people, probably 80%, just said, "It's so incredibly easy." I just grab that and tomorrow morning I got something running."

So community building is both about advancement and evolution, but it's also just about incredible convenience, like just make my life easy because of this project, right? I don't know necessarily that the blockchain thing has that attribute yet. It doesn't necessarily make your life easier to do X, and that's where I think still some of the differences are.

[00:54:04] JM: You're on the board of Aurora. Do you have any predictions for what parts of the self- driving car industry will be open source?

[00:54:13] MV: There are some open source elements out there for self-driving cars that people can start off of. I think, right now, and certainly whether it's database systems, some of the AI components that are in their, there are some open source parts to it. I think, right now, a lot of what's happening is the very early stages of creation of the system and people are really worried about open or close source. They're just sort of like running as hard as they can to make it work.

I think that once we sort of establish the basic architecture of how a self-driving system might work, I think we might see some components of it start to open source. But I think, right now, they're grabbing some open source parts from it here and there to kind of construct the vehicle that runs. They’re not really worried about open sourcing their thing, because they’re just so heads down to make it work. Eventually, we might see something.

[00:55:12] JM: And one tension that I see in the software/hardware architecture model for the self- driving cars that, on one hand, you want the self-driving car to be like a data center, because it has to collect so much data. You want it to collect so much data. You want it to be able to process. You could probably allocate an entire data center's worth of resources to a single self-driving car if you somehow had that available, and yet you also, to some degree, want it to be like a smartphone. You want it to be light and not have that much going on, because it's ultimately a vehicle.

Have you different, I guess, hardware/software architectures that are evaluating a tradeoff and what trends or predictions do you have?

[00:55:57] MV: Well, there's all sorts of architectures. There’s a huge range of them. You have to look at it though from the problem of the physical world that you're trying to solve, which is that a car is an object that moves around every day and is constantly making life-and-death decisions. My smartphone doesn't really make life-and-death decisions point in time for me, but a car does.

So you need to make sure that you have enough computing capability in that vehicle, if it's autonomous, to make a life-and-death decision that is at least comparable to what a human being does today when they’re driving that vehicle, right? So you have to have at least that level of competence in the vehicle itself.

Because humans are pretty bright and pretty capable, and we have a lot of good sensors, like eyes, and ears, and touch, and so forth, you're going to have that level of compute in the vehicle for some time, and then everything else can be offloaded to the cloud.

So I think, sort of today, smartphone capability is just not enough. I'm also not a real believer in like tele-operations, where you basically just have a camera in the car and like everything else goes up to the cloud, because of the latency and delays that exists in that use case. So I think what you're going to see is vehicles, cars, with a fair amount of intelligence built into them for some period of time precisely because they're facing that life-and-death decision at any moment in time.

If you didn't have to do that, you could certainly shift a lot of the compute to the cloud and you can make them much, much simpler, but I don't think that's really realistic in the very near term, given the importance of those decisions that the cars are making.

[00:57:46] JM: Yeah. So this model you alluded to where you have, let's say, a self-driving car or a self-driving truck and if it encounters a situation that's confusing, you phone home and you have a tell operator who is remotely operating the truck and they take over. There are entire businesses that are built around the thesis that that can work, but it seems like it's a network partition away from fatal errors. How do people resolve that?

[00:58:17] MV: Look, I spent 14 years at Cisco. I'm a networking guy, and I think I have a moderate understanding of how networks, best effort networks, work. If this thing – If communications work over a TCP/IP, you have twin problems with it. One is that you don't know what route the packets take, and so they can take longer, they can get dropped, they can run into congestion. So you can't really guarantee that any given packet gets from A to B, nor can you guarantee the time which it takes for that to go.

The action of braking in a car takes a human being about a second, maybe just under a second, to put your foot on the brake. If it so happens that my zoom conference down the street is slowing down that car's camera view into the tell operator by half a second, I've increased my chances of an accident by a lot, because essentially the travel time just went down from – Like my braking time went from one second to one and a half seconds, and if you're at a certain velocity, that's a lot of yards, and that could be life or death.

So I think when you're dealing with human life, you can't sort of make these sort of hand wavy assumptions that, "Oh! Don't worry. The network will be fast," because I think all of us are on video calls all the time. We know what that happens. I mean, even the good ones have moments where it stops working or it works a little less well, the image get blurry. So I am very suspect of systems that say, “Well, we’ll let the tell operator take over,” at least in today's driving contest. I think that you have to either have a physical driver that's a safety driver or you have to have enough intelligence where the vehicle can self-operate in the event that the network isn’t providing the sufficient latency or visibility to a driver that's remote.

[01:00:12] JM: All right, last question. What's something you believe about Venture Capital that most venture capitalists disagree with you on?

[01:00:19] MV: Look, I put it this way. The venture capital business has been going through a 15-year transformation, I think, from being kind of a cottage industry into a very professionalized industry. You see more segmentation in terms of stages of investing. You're seeing venture capital firms offer an enormous amount of support services, whether it’s helping companies sell their product, or recruit their employees, or market themselves, or fundraiser some more. There is a venture funds that focuses on specific sectors. There are all sorts of sort of segmentation and professionalization of the industry. The industry has become much more analytical than it used to be. It's a very data-driven compared to the way it used to be. So the notion is that this old sort of craftsman-like cottage industry business is going to go away.

I tend to disagree with that. I think that at the end of the day, venture capital done right is still ultimately a relationship between a couple of founders and one or two venture capitalists who are their partners, and the job of the venture capitalist is to be an expert and a domain expert and may be a consultant, an advisor, but also just to be a real human being, because the job of being a CEO, the job of being a founder, can be very lonely and often times you're wandering in an area you have no idea on what you're doing.

Having somebody that you can rely on and call on and just be sort of an all-around coach for you is very, very important, and will remain important for very long period of time. So I think even though this industry is professionalizing and there is a thesis, some people are even saying like, “Oh, well. Just drop your data into a machine learning algorithm and we’ll decide whether we’ll invest or not.” I think there is to be a part of industry that looks like that, but what founders will find most valuable is still that first investor that truly becomes a multi-year partner for them. I firmly believe that that's real venture capital

[01:02:16] JM: Mike, thanks for coming on the show. It’s been great talking.

[01:02:18] MV: It’s a pleasure.

[END OF INTERVIEW]

[01:02:23] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code’s site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]