EPISODE 865

[INTRODUCTION]

[00:00:00] JM: Open source software allows developers to take code from the internet and modify it for their own use. Open source has allowed innovation to occur on a massive scale. Today, open source software powers are consumer client applications and our backend cloud server infrastructure. Linux powers single node operating systems, and Kubernetes is the foundation for new distributed systems. Hadoop created an open source distributed file system. Spark gave us a computational run time on top of that file system, and Kafka created a middleware platform for shuttling data from one place to another. There are numerous other examples of how open source has changed the world of software development. Open source has also reshaped the business landscape of infrastructure software companies.

A common business structure for a modern infrastructure company is the open core model. An open court company maintains an open source project that is free to use, but also sells a product or service around that project. Companies with an open core model include Red Hat, HashiCorp and GitLab. Many companies are building a thriving business with the open core model, but these companies do not directly control the most important part of the infrastructure supply chain; the cloud provider.

Cloud providers have a fundamental tension with open core companies, because the cloud providers offer services that compete with open core companies. In addition to the issue of cloud providers competing directly with the open core companies, some people have questioned whether Amazon Web Services is capturing an unfair portion of the value that is being created by open source.

Amazon Web Services is the biggest cloud provider and it has built a large catalog of services that are built off of open source software, but AWS has not historically contributed heavily to open source relative to the value that it has captured. One example of an open core company, which has lost market share to an AWS cloud hosted offering is Elastic. The open core company which maintains the Elasticsearch open source project.

Amazon Elasticsearch service is a closed source hosted offering built on top of Elasticsearch. Elastic, the company, has increasingly intermingled proprietary software with their open source repository, making it less clear how that open source repository can be used by companies that want to deploy it for their commercial use. But maybe that's a fair defense. Maybe it's just what's necessary to defend Elastic from AWS, otherwise capturing the lion's share of the market of enterprise customers who want Elasticsearch product and support.

Open core companies such as MongoDB, Redis Labs and Cockroach Labs have responded to the competitive pressures of AWS by changing their licenses and making it more expensive for cloud providers to offer a cloud hosted offering of their open source project. The dynamics between cloud providers and open core companies will continue to evolve in the coming years. The norms around open source are up for debate.

Joseph Jacks is the founder of OSS capital, a venture firm focused on investments in commercial open source software companies. Joe returns to the show to discuss the changing landscape of open core companies and the benefits of permissionless innovation.

If you want to find all of our old episodes about open source software, or about Elasticsearch, or about cloud providers, you can download the Software Daily app for iOS. It has all 1000+ of our old episodes. It might be up to 1100 at this point. It has related links, greatest hits, topics. You can comment on episodes and have discussions with other members of the community. If you want to skip the ads, we have a paid option for ad-free episodes. You can go to softwareengineeringdaily.com/subscribe to support us with $10 a month or $100 a year. We'd much appreciate that.

This app was built largely by Altalogy, which is a company that has been developing much of the software for the softwaredaily.com website, which is another way that you can view all of our old episodes, and Altalogy has done a great job. We also have the Android app coming out soon, if you're looking for that.

With that, let's get on to today's show.

[SPONSOR MESSAGE]

[00:05:11] JM: DigitalOcean is a simple, developer friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow.

DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support, plus they've got 2,000+ tutorials to help you stay up-to-date with the latest open source software and languages and frameworks. You can get started on DigitalOcean for free at do.co/sedaily.

One thing that makes DigitalOcean special is they're really interested in long-term developer productivity, and I remember one particular example of this when I found a tutorial in DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence, and an attention to just getting developers off the ground faster, and they've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That's the D-O.C-O/sedaily. You will get started for free with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:07:12] JM: Joseph Jacks, welcome back Software Engineering Daily.

[00:07:14] JJ: Thanks for having me, Jeff.

[00:07:15] JM: We're going to talk about open source today, and open source business models and the relationship of open source companies to cloud providers as well as your long-term vision for where open source is going. So, hopefully we can touch on all those things.

I want to start with the very concrete example. Let's say I have a database. I have JeffDB. It's an open source database. It came out of Software Engineering Daily. This is a database we made to host podcasts, totally hypothetical.

Let's say we open source this database. People start using it. They start running it themselves. We say, “Okay. We’re going to make JeffDB. We’re going to make a company, and this company is going to offer support and services around JeffDB.” We start up this company. It gets successful, and now we say, “Okay. As our next product, we want to have a cloud hosted offering of JeffDB.” What should our go-to-market strategy be for creating and releasing that cloud hosted offering?

[00:08:22] JJ: Okay. So, it depends on what the acceptable preconditions for that goal are. So, what I mean by that is you asked how would you most effectively create a hosted cloud service for JeffDB. What you didn't really cover, which I think is really important as far as open source projects are concerned in relation to building companies and sort of a successful business. This is sort of what I call commercial open source fusion of commercialization and open source in sort of one phrase. This is COSS, the COSS term, commercial open source software.

What I believe is the biggest precondition to ensuring that an open source project is very successful as the basis for a product or a hosted offering or some type of platform is very large-scale, widespread adoption of the project, community participation, industry standardization, industry acceptance of the technology, external contribution.

So, widespread adaption means a lot of different things in different contexts. So it that means building a project that is attractive to external developer contribution that is attractive to other providers or other platforms to integrate with it, support it. That is attractive to technology trends and paradigms referring to the sort of architectural principles and the patterns that are embodied in your project as sort of things to aspire to.

Particularly at the database layer for JeffDB, hypothetically, you presumably would be building something that is very differentiated on a few dimensions, whether it's combining or integrating previously disjointed sort of abstractions or paradigms into one. Making performance better in some set of dimensions, making consistency better, availability or whatever.

I would say that the biggest precondition to a large successful hosted open source project as a service would be building a lot of community adaption, and that encompasses not just usage of the software, but contribution to the software and all those other things that I mentioned. This doesn't mean that you can't build a fully proprietary hosted database technology offering and build a large business. It just means that the tradeoffs in terms of going down that path are very different. They tend to be far more capital-intensive, far more costly in terms of development of IP and a bunch of other things. But that wasn't quite your question.

I can answer your question from the perspective of the many commercial open source software companies that have evolved toward heavily investing more into hosted cloud-based services of their open source projects. So, companies like MongoDB, and Redis, and Influx Data, and Cockroach Labs and so on. Each of those companies has sort of evolved in different ways and made different decisions in different tradeoffs in terms of building and investing in a hosted as a service offering of their open core, their open source project. I can talk a little bit about some of the paths and approaches there.

My own views are definitely influenced by kind of what has happened over the last five or six years and then also what sort of governing principles are sort of setting up open source projects for what I view as those preconditions for success in terms of widespread adoption, widespread external contribution, industry standardization, distribution and so on. At a high level, it's a simple question, but there's a lot of nuance to the context.

[00:12:01] JM: And the reason I ask this question is because we have seen this subtle debate that has turned into a philosophical debate. The subtle debate of should open source companies that feel threatened by cloud providers change their licenses in order to make their businesses more defensible from those cloud providers. There's a question about business viability. Does changing our license make our business more viable? There're also questions about the “spirit” of open source.

So, for example, you have said if AWS truly cared about respecting the fundamentals of open source software innovation, they would open-source their proprietary control plane, all data

services, and partner with commercial open source software vendors instead of treating them with hostility. So in this tweet, you said the fundamentals of open source software innovation.

What I don't understand is what are those fundamentals? I mean, open source software is this thing that has emerged organically. Who has ordained any kind of fundamentals?

[00:13:12] JJ: Okay. So, this is a really great question, and thank you for pointing up. Most of my tweets are sort of off-the-cuff and require very extensive elaboration and unpacking ideally to a blog post or this kind of setting. So, I really appreciate you having me on to share a lot and unzipping.

[00:13:28] JM: We're decompressing. We're unzipping.

[00:13:29] JJ: Unzipping and – Well, yeah, depends on context. Yeah, unpacking, elaborating. So, you asked a really, really important question that I think a lot of people don't consider or think about enough. So, the fundamentals of open source software innovation are basically traced back to this very polarizing, but in my mind, super smart individual named Richard Stallman. Richard Stallman was AI, computer science visionary very early on in the MIT. I think predecessor to the CSAIL, computer science and AI lab.

In the early 80s, just as the software industry was, you could say, catalyzing, shortly after Bill Gates and his letter to hobbyists where he basically made the case for charging for software before charging for software was a thing and proprietary software was sort of not really invented yet. Richard was motivated very shortly after that letter that Bill wrote in, I believe, 1976. I could be wrong though. To create both the Free Software Foundation and the principles and the sort of governing what Richard called and still calls the four fundamental freedoms.

I might be getting them in the wrong order, but the four freedoms can be sort of boiled down to and distilled into four verbs, that in order to be accepted as sort of meeting the definition that Richard sort of pioneered and proposed for free software, which eventually evolved into open source software in 1998, and can get into that. The four freedoms are; unlimited, unrestricted, what I would sort of paraphrase as ability. Richard would very adamantly say freedom to see the source code, run the source code, modify the source code and distribute the source code.

So, those four verbs are those four freedoms or abilities effectively have unrestricted, untethered parameters. So, in what Richard implemented as far as a sort of model for ensuring adherence to those four principles, or freedoms, or abilities, is still today, after several iterations known as the GNU Public License, or the GPL.

The GPL went through – And this is something that we could probably cover over two or three podcasts with much more detail that listeners or, really, I think anyone should torture themselves in going through it and understanding. But GPL 1 and 2 ultimately sort of had their various issues and contentions on different dimensions and the industry sort of settled on GPL 3 for a variety of reasons in the cases where the GPL 3 is used over Apache 2 and BSD and MIT and other licenses.

So, to kind of circle back to answer your question at a high level, the fundamentals of open source software innovation I believe, in my view, trace back to Richard Stallman coining, creating and pioneering and sort of you could say instigating the four fundamental software freedoms as he calls them, what I call them as abilities or capabilities codified in unlimited, unrestricted, seeing, running, modifying and distributing of source code.

Fast forward from 1983 to today, 35+ years later, what we've seen is a really big evolution in the framing of those four fundamental verbs in the context of software development and also in the philosophical orientation of their implications. Not to kind of self-promote my own blogs or anything, but like I wrote a few thousand words on the sort of permissionless software innovation movements as I call them. That kind of cover the philosophical orientation and motivations that drove Richard Stallman to define those four freedoms in 1983, and getting to open-source what the philosophical orientation and evolutions were that cause the open-source definition to emerge and the open-source term and movement, which occurred in 1998, about 15 years after Richard Stallman pioneered the Free Software Movement.

Coming to today, about 15 years later, so from 1998 to 2013, 2014ish. I'd say actually shortly before Kubernetes came out. What I believe is we're entering a third era. So if you look at the first era of sort of, let's say, permissionless software innovation, more abstractly, taking the term

open source out of the framing. First era was free software. Second era was open source. I think what the third era looks like is a sort of commercial open source dynamic.

So, just to kind of give you a philosophical kind of evolutionary layering on that, the first era. Again, I sort of written about this in a little table, but Richard was very heavily motivated by liberalistic sort of aspects of events of the times and had a highly negative and antagonistic relationship with a proprietary software. So, in fact, he did and still does view all proprietary software as evil, specifically because proprietary software does not adhere to any of the four freedoms, right?

The second era is sort of philosophically involved away from this sort of hardcore liberalism and more toward utilitarianism and pragmatism. So, the sort of creators and you could instigators of the open source software movement were really building on the shoulders of Stallman's principles and sort of certainly not discarding them, but more refraining and kind of rearticulating them in the sort of era of Linux development and the LAMP stack and distributed development and sort of *The Cathedral and the Bazaar* principles that Eric Raymond wrote about, which is a great book. I encourage people to read *Cathedral and The Bazaar*. Talks about that the cathedral model being very concentrated, centralized software development methodology, and the bazaar being decentralized, sort of knowledge of the crowds. Of course, you can have a cathedral within a bazaar, bazaar within a cathedral and uses all kinds of multivariate ways of combining those two things, but they are fundamentally two different types of approaches.

The second era, they're open source, utilitarian, pragmatic. The relationship to proprietary software in that second era was not antagonistic, was not all proprietary software as evil. What I believe that third era sort of reflected in terms of its relationship to proprietary software was that proprietary software was not necessarily evil.

What I think the third era looks like is as follows. We don't really have one group of people or sort of one person that has sort of instigated third era. I think the instigators are effectively this critical mass of very large, successful companies that have fundamentally built their businesses around open source software, and open-source software projects. I call them commercial open source software companies.

I think the third era can be described as the commercial open source software era. The philosophical orientation of the third era is very capitalistic, but sort of what I would like to sort of propose as a type of meritocratic capitalism. So, isn't viewed as this sort of monopolistic zero-sum, Peter Thiel-oriented winner-take-all capitalism, but I would say as sort of positive-sum meritocratic capitalism I think is one way to describe that third era.

The third era from its relationship to proprietary software would be more of one that is integrated and complementary. So proprietary software and the relationship to commercial open source software is certainly not evil as with the first era. It's certainly not necessarily evil as with the second era. It's actually complementary, integrated and fused into the sort of development model kind of combined.

So, that's maybe like a long-winded segue into I think a lot of these sort of confused and very sort of gradually evolutionary learnings that we're going through in this sort of current world that is very dominated increasingly by open source software innovation. The kinds of questions that are starting to come up are, "What is open source really mean? What is the open source definition really mean? Does anyone kind of control that open source definition? What are the kind of constituent parts of company building that start to make sense? How can we solve the sustainability problem and the context of company building?"

[00:22:01] JM: Okay. Hold on. Before we creep into other areas of this discussion, I want to focus on this question of open source fundamentals. Because what I hear in what you're saying is that there has been a lineage from the ideas Richard Stallman to today and that we should pay some respect to the norms that were established in those early days of Richard Stallman. Whether or not that accurately captures what you're saying.

What I would argue is that we also owe a due respect to what AWS has done. We owe it to AWS to allow AWS to capture a large percentage of commercial open source value, because they provided the necessary startup energy to make the cloud a reality. So, what I see here is two –

[00:23:16] JJ: I completely agree with you, by the way, on that.

[00:23:18] JM: But this is where the tension is, is to fully respect the “fundamentals” of open source, which by the way are not codified in any kind of legal structure, or contract that the community as a whole has agreed to abide by.

[00:23:35] JJ: Well, Richard would agree that they are codified in the GPL license, but I hear what you’re saying.

[00:23:39] JM: That's fine. I mean, I can write a shopping list and I can say I have codified my shopping list. Nobody else has agreed to buy into that shopping list.

[SPONSOR MESSAGE]

[00:23:39] JM: Mux is an API for video. Mux makes beautiful video possible for every development team. Post a video, get back a video URL that plays back on any device in just seconds. Video is not easy to manage. There are bit rate ladders and CDN decisions. The decisions about how to serve your video depend on the desired quality and reliability and speed.

Mux allows teams of all sizes to get up and running in minutes, and live video is just as easy. With a single API call, you can get an endpoint where you can push a live video stream and you get an ID to playback the stream.

Whether you have a specific idea for a video application or you just want to tinker and get creative, Mux is a tool that makes amazing video experiences simple. You can listen back to our previous episodes with the Mux team to find out why video is a complex engineering problem.

Mux was founded by experts in online video, including the creators of the biggest open source video player on the web, which is Video.js Mux is trusted by leading providers of online video, like CBS, and Vimeo. You can sign up for a free account at mux.com and get $20 in free credit to get started. That's Mux, M-U-X.com. You got to love the three letter domain name. Mux.com. Get your free $20 in credit and make something cool using video.

[INTERVIEW CONTINUED]

[00:25:46] JM: his is what I think is so confusing about this whole argument is that people talk about it like there are norms. People talk about it like there are established laws that we need to abide by. But, all I see is a set of constituencies who have their personal motivations, like what is Richard Stallman motivated by? I don't know. He's got his own motivations. But, he has his own selfish motivations that led him to create those ordinances, and we all have our own selfish motivations.

What I see is what is bubbling up today is that there is a fundamental tension between what AWS has earned and what the open-source community and the open source vendor model has "earned". So, why do the open source vendors – Why did they end up changing their licenses? This is what's so confusing to me.

[00:26:41] JJ: Here's why. I think I know enough to answer that question in a general way, and I don't say that frequently [inaudible 00:26:46].

[00:26:47] JM: By the way. In your answer, please first frame the different open source companies that are changing their licenses and give the justifications that they have gave for changing their licenses.

[00:26:58] JJ: Sure. Most of the companies – So, I will list a litany of specific examples. Before doing that, I'll say most, if not all the companies that have decided to change the licensing model of either the core open source project that the company is fundamentally based on, or introducing a separate license for additional projects that are peripheral, integrated with or complementary to their core open source project. They're doing those things for a fairly common set of reasons, which you alluded to, which is the fear that cloud providers will have an unfair competitive edge or advantage in terms of capturing value that they're effectively – That is unfair or unequal to the efforts that produced the software in the first place. The overall energy output that produced the software, which is predominantly coming from these companies.

Okay. So, there's a lot of nuance to that, like sort of broad preface, and I can kind of impact that a little bit further. The specific examples of companies that have sort of done this are companies like Redis, MongoDB, Confluent, Cockroach Labs more recently, MariaDB, actually a few years

back, and a number of other companies. Interestingly, this would be pretty relevant to your opening question, JeffDB. Those are actually mostly database companies.

So, here in lies the segue to maybe some commentary on this, which I hope doesn't get me in trouble, because I know the founders of all those companies, and I'm pretty honored to have a lot of them speaking at a conference we're putting together pretty soon called the Open Core Summit, which I can talk about a little bit more later.

My personal view is that these licenses are vastly unnecessary, motivated by effectively a non-problem. So there's sort of a non-solutions to a non-problem. I also believe that as history plays out, we will observe that these licenses create a slippery slope back toward basically the sort of traditional, fully proprietary software development model.

[00:29:11] JM: Windows 95.

[00:29:12] JM: And also do more harm than good on a few dimensions. So, all of the conversation around this unfortunately is very Pythian and high-level, and so I'm glad to have the opportunity to talk at length on a lot of these things here with you, because it's really impossible to sort of like kind of describe the nuance over a tweet or even, frankly, over a blog post. So, really this is actually why we're doing a big conference on this. Not because of the specific issue, but because of a much broader class of fundamental things.

So, you actually mentioned something that I want to go back to, which is that Amazon should be given a lot of credit for sort of creating and galvanizing this cloud computing movement. Obviously, investing –You didn't say, but like tens of billions in capex to build all the infrastructure, lay down the rails and basically build the upfront investment and the sort of building blocks for superefficient consumption of software and delivery of software.

Now, what I want to go back to is the sort of like the fundamental open source software development model for a sec. If you look back to those four freedoms that Stallman kind of laid down 35 years ago, they've really stood the test of time and been respected and implemented in the sort of litany of open source licenses that are used across the industry today in all kinds of

cases. Whether it's MIT, or BSD, or even in a lot of cases, the GPL 3 license. Of course, the Apache 2 license and so on.

What those licenses basically enable is a positive-sum value creation and value capture dynamic, right? So, there's quite a bit to unpack there. What we see with the inverse, which is a zero-sum value creation, value capture dynamic, is proprietary software. Zero-sum value creation, value capture, is a company that ships a fully proprietary product and is the primary source and driver for creating the value in the world and universe that is enabled by that software or that product and is also the primary entity capturing the majority of the value as it relates to a payment exchange between the consumer and user of the platform and the software and that entity.

So, for example, the consumer and user and entity of Slack, certainly capturing lots of value as the end user, because they're more efficiently. They send less email and their teams can communicate more effectively in this new innovative way through the Slack messaging application. In terms of capturing value though, that part of the value capture dimension isn't really reflected by or sort of like measured by the traditional economic model.

What is reflected is Slack's market capitalization and Slack's revenue. So, as a company that is just about to go public, it's actually not a bad example. Slack will probably be worth between $10 and $20 billion. There simply is no other company that is capturing value on the Slack platform commercializing Slack that is capturing anywhere near that amount of value. So, that's a zero-sum value capture dynamic or zero-sum value creation and value capture dynamic.

Conversely, let's look at a company like – I'll pick Mattermost, is a really good example. So, full disclosure, OSS Capital is an investor in Mattermost and Go Mattermost, but Matamoros is –

[00:32:34] JJ: The open source Slack company.

[00:32:35] JJ: Mattermost is commercial open source Slack, but they're also just a new kind of innovative enterprise messaging platform. They've got a really great developer. Sort of friendly set of APIs and ecosystem tooling and SDKs, and they're growing really quickly and they're doing really well.

This is an example of a company that is a positive-sum value creation and value capture company. So, you can take the Mattermost codebase. You can embed it in your application and commercialize that application. It's like a domain specific product. You can see, you can run.

[00:33:06] JM: I can make Slack for farmers.

[00:33:07] JJ: You can see, run, modify and distribute the Mattermost core runtime codebase without restriction. So, that's a commercial open source company, and they're doing really well. They've built enterprise business.

[00:33:18] JM: Just to clarify what you're saying there. So I can make Slack for farmers. I can make Slack for taxi drivers. I can make my own product based off of the codebase as supposed to Slack where I don't know –

[00:33:30] JJ: Closed ecosystem.

[00:33:31] JM: Closed ecosystem.

[00:33:32] JJ: Proprietary. Okay. So here's my comment going back to cloud providers in full.

[00:33:35] JM: Please. I know. I know you're long with that argument. Understood.

[00:33:39] JJ: Let's get back to database. The fundamental reason why these new, what I like to call them as is discriminatory or non-compete licenses. I'm still oscillating between calling them either discriminatory or non-compete licenses. They have to fall somewhere in the middle of those two terms, but let's just call them discriminatory licenses. This is probably maybe a slightly sort of jabby term, but generally speaking I think it fits.

The reason those licenses are a bad idea is – And so with all the context, is because what they do is they is they actually discriminate and prevent value creation and value capture.

[00:34:19] JM: By the way, you're talking about the licenses of MongoDB, Elastic.

[00:34:21] JJ: I'm talking about variants of BSL, SSPL and effectively new type of things – This is like a third way of describing them, pseudo-open-source licenses.

[00:34:32] JM: These are licenses that impose –

[00:34:33] JJ: But these are licenses that really look like proprietary licenses. Effectively, you can do everything, except if you want to do one specific thing with the software. But that one specific thing happens to be a very broad class of value capture sort of approaches, then you're prevented.

So, for example, most of the licenses are oriented around if you want to take the software and charge for it, or host it up as a service, or build a cloud offering, and you are anyone. Not necessarily a cloud provider, but anyone who wants to do that. You have to pay the company a license. Everything else with that sort of traditional open source idea is still there. So you can still see the source code in most cases. You can still run the source code anywhere you like in most cases. You can still modify and submit changes to the source code in most cases.

In terms of the distribute verb, in terms of distributing and actually charging someone to pay for access to the software over an API over a hosted service. That's where the discrimination starts to come in. Here's the main contention I have with that. The main contention is that in this modern era of software getting consumed over cloud APIs and cloud services and cloud computing is certainly a really big thing. I believe that there's a lot of conflation of that distribution model and that sort of consumption model with a business model.

I actually don't think – This is maybe more controversial that I should sort of say, for now, but I might regret saying this, but like I don't think that the as a service hosted cloud-based approach to distributing software from the creator side and consuming software from the end-user side is a business model. I believe it is exactly that. It's a distribution model from the creator side and it is a consumption model from the consumer end-user side.

So, when these commercial open source companies have sort of concluded that there is an existential threat or fear of imbalanced competition from cloud providers being able to take their

projects, post them up as services and capture more value in them. I think that there are a few fundamental flaws in that, and I'll outline them.

The first fundamental flaw is that because open source software operates under this positive sum dynamic fundamentally with slight unrestricted see, run, modify and distribute of the software. The moment you start changing one of those abilities or those – Going back to Stallman, and I try not to use this, because it gets very politically heated, freedoms. You really start to very quickly deviate away from the sort of properties that enable all the goodness and all of the innovation potential of this approach of creating and innovating things permissionlessly. You noticed I didn't say open source or free software. It's just permissionless innovation.

[00:37:34] JM: I can interject with an example here?

[00:37:36] JJ: Sure.

[00:37:37] JM: Okay. So I interviewed render.com last week. Do you know what rendered.com is?

[00:37:42] JJ: I think – Refresh my –

[00:37:44] JM: It's a new layer two cloud provider. So, they just came out of Stealth recently. Started by the former head of risk at Stripe. He was like employee number eight at Stripe. He left Stripe. He started this layer two cloud provider. Looks really nice. Haven't tried it yet. It looks great. Render.com. I am very interested in trying it out myself. Kind of like a next generation Heroku sort of thing.

But I was looking at their services that they offered and like they have opinionated ways of deploying things like Docker containers and MySQL databases. This is a startup, right? They've raised a couple of million dollars, but they don't have infinite money.

As far as I know, they would be restricted from offering Elasticsearch, or MongoDB, or Kafka by these companies. No?

[00:38:33] JJ: Okay. So, again, the nuance here is really deep. So, you mentioned three projects that –

[00:38:39] JM: Yeah. These are companies that have changed their licenses.

[00:38:41] JJ: Correct, but they're all doing so slightly differently. So, I didn't get a chance to articulate like –

[00:38:47] JM: Okay. But just broadly speaking –

[00:38:49] JJ: So, Kafka is still governed under the Apache 2 license.

[00:38:52] JM: But do you understand my broad point here?

[00:38:53] JJ: I do. Yes.

[00:38:54] JM: We have a startup that is being inhibited from building the business of what we should expect out of a cloud provider. I should be able to get hosted Kafka from my cloud provider. If I can get up-to-date open source quality hosted Kafka, then it really – From a new startup cloud provider, it really calls into question, "Is this open source anymore?"

[00:39:17] JJ: It's not, but Kafka is a bad example, because Kafka is still governed under the Apache 2 license, and Confluent actually –

[00:39:22] JM: Okay. Kafka is a bad example. What's a good example?

[00:39:24] JJ: Confluent's approach here I admire.

[00:39:27] JM: Confluent, the Kafka company.

[00:39:29] JJ: Confluent the company that the founders and people running the company, Jay Kreps, and Neha Narkhede, and Jun Rao and others created the Kafka project at LinkedIn and they left LinkedIn and started Confluent. So, yeah, the creators of Kafka are effectively running

Confluent, and that's a company that is commercializing Kafka and building a platform around sort of a lot of paradigms and things you can do with Kafka.

What they did with the licensing model was a little bit different. In fact, each one of these companies has sort of implement their own approach. So, there is certainly some generalization that you can make here around a lot of the motivations driving these companies. In some cases, a few of these companies have common investors. So I think there's a linkage that you can make there. However, I'm not going to go down that rabbit hole.

I think the distinction with –

[00:40:13] JM: I spoke with them on the show.

[00:40:13] JJ: With Confluent – Yeah, I think you're referring to Mike Volpi. He's a Michael and a great person. I would say Confluent's approach is a little more nuanced. So, what Confluent did is they decided, "We're not to change the license of the core project, Kafka." So, Kafka is still Apache 2 licensed.

What they did was there's an ecosystem of value-added, domain-optimized and specific kind of control plane query layer analytical and sort of peripheral complementary tools that Confluent has built that deeply integrate with Kafka and sort of complement Kafka, and they've provided the Confluent enterprise license or Confluent community license. I forget what they call it, but it's a new kind of license that creates certain restrictions around the usage of that software.

[00:41:04] JM: Like KSQL or something?

[00:41:05] JJ: Yes, and the other projects, but still allow the community to see the source codes of their resource available licenses, and I believe you can contribute code and a few other things. But there's sort of like a hybrid between certain permissiveness and capabilities or freedoms if you will and a balance of things that ensure Confluent's commercial strategies can sort of be enabled most effectively.

But what they've done smartly, I believe, is they've sort of de-coupled sort of, say, three things. They've decoupled the core open source software that is creating the vast majority of the value. So, the exponent of the maximal value creation for Confluent is the Kafka project.

The second layer of licensing is applied to projects that sort of surround Kafka that are sort of complementary and integrate with that core project. There are certain restrictions there, but for a lot of intents and purposes, you could sort of view those projects as sort of pseudo-open-source, but they're not really fully open-source. There're quite a lot of political argument over like if you remove one of those abilities, you no longer open source at all. But the sort of like medium or sort of hybrid model that they've chosen certainly appeases lots of developers.

The third layer of licensing there is fully proprietary enterprise licensing, where you have to pay Confluent to see the source, run the source, distribute it, or no one really wants to modify or distribute the source they paid for.

[00:42:39] JM: Okay. You made your point that Kafka is a bad example. Can we find a good example from the ones I mentioned?

[00:42:44] JJ: No. I'm saying Confluent's approach to sort of layering three types of license models that are applied to different sort of parts of their stack is actually smart. What I don't think is very smart is when a commercial open source software company that has a successful open source project and an enterprise business as well, so like a proprietary. I call this the crust around the open core. So they've got a proprietary set of control planes or a proprietary hosted service or a proprietary set of extensions or additional features.

When a company like that mid-swing of growing their community and ecosystem decides to change the entire license model of the core open source project, which is the source of creating most of the value in the company and they do that either very later sort of into their sort of evolutionary model, like with MongoDB or with sort of companies like, you could say, MariaDB, or Cockroach Labs or other companies that do a little bit earlier on. I think that the dangers there are that you scare off a lot of external contribution. You put your community sort of governance dynamics at risk of a certain level of integrity that otherwise would be pretty easy to uphold as

long as you maintain a level of honesty and transparency in alignment with your ecosystem and, potentially, a lot of other derivative issues.

With MongoDB though, many millions of people use MongoDB, and when MongoDB released the SSPL, in terms of measuring different things that we can measure, growth of the company, employees headcount, revenue, and this is a public company. So we can actually measure a lot of these things. We can't really see any negative effect that it's had. In terms of the governance of the code base and the contributors to the codebase, we also can't really see any negative effects, because the vast majority of contributors to MongoDB actually are employed by MongoDB.

However, when you have certain open source projects that are governed and controlled by a company more democratically in a more decentralized way, say, half of the core committers work at the company, maybe only 10%, and the company still been able to build a large business. That's when you start to run into some really potentially dangerous areas in changing –

[00:45:00] JM: What's an example of a project?

[00:45:02] JJ: I don't think of any immediate examples off the top of my head, but I think that there's sort of spectrum and a gradient of control over how that could play out. I'll give you one potential example. DataStax is the company behind Apache Cassandra. DataStax has not changed. I'm literally inventing this on-the-fly for you, Jeff. So this is just hypothetical.

The Cassandra project is not fully controlled by DataStax, and its run effectively by PMC chair members and committers at the Apache Software Foundation. I think there's a few that do work at DataStax, but when DataStax was founded and started years ago, I think close to 10 years ago, the creators worked elsewhere and the company was founded just by really passionate evangelists and people who are excited about the Cassandra project.

Now, DataStax is a successful business. They're a hundred million revenue business. I think they're on their way to either an IPO or a large outcome. They're one of the companies that

we've been looking at and measuring and evaluating among 40 others for close to five years now in our commercial open source software company index.

So there are examples of companies that don't actually fully control and dominate the contributor base, the governance model and the ecosystem of the projects the companies are based on. So, because you asked the question, I'm just kind of, again, framing this is a hypothetical. If those companies somehow were able to and successful in changing the fundamental license model governing the project that they're based on, the consequences and the effects would be really dramatic and negative I believe.

So, I guess maybe philosophically, but I think this plays into the most optimal long-term meritocraticly, capitalistic model as well, is you should decouple – As a commercial open-source software company, I believe you should always appreciate that the software a company is producing that is creating the most value for the company is decoupled from the company, and the company should actually focus on building a large successful business with certain zero-sum dynamics, but the project itself that creates a lot of the value for the company should respect and incorporate and implement positive-sum dynamics and enable a large constituency and a large ecosystem around it to be successful. That includes cloud providers. That includes anyone who wants to run as a service. That includes anyone who wants to commercialize.

I believe the most successful open source projects will always adhere to that. So we have projects like Kubernetes, projects like Linux. Certainly, I have seen super large successful projects across all kinds of dimensions that are fully decoupled from the company create large economic opportunities for many constituents for companies in the industry that are certainly not viewed as tech platform companies, like banking, retail, insurance, government. We've other cloud providers be able to take projects and incorporate them. Certainly, lots of commercial open source competitors.

I think the largest, most successful ecosystems over time will be constructed based on fully permissionless and highly permissive governance constructs at the project level. But at the company level, certainly, you have a hierarchical cap table. You have a hierarchical management team effectively. The founders and the people who join after the founders and the companies looking to build as much of a differentiated product experience as possible.

What I believe though is there are certain fundamentals of commercial open source companies that require business model construction, economic optimizations, product development and all the things that sort of allow those companies to be large and successful. Those things are fundamentally different as compared to proprietary software companies, and they need to be measured and evaluated fundamentally differently.

So, again, it's a sort of longer conversation, a rant. But if you look at proprietary software companies just to kind of close this out, the software is tightly coupled to the company. With commercial open source software companies, the software is completely decoupled from the company. Software that is creating the vast majority of value is decoupled from the company.

I think the biggest question that the technology industry is facing now, because this is a huge category. We started measuring these five or six years ago. It was less than 10 billion and Red Hat was 90 plus percent of that value. Now it's 130, 140 billion and Red Hat is less than – Well, basically –z

[00:49:36] JM: This is your index of commercial open source companies.

[00:49:39] JJ: Right. Depending on how you measure the full universe between 10% and 20% of that value. So, it's a very large, fast-growing universe that is not immaterial and insignificant, even compared to the size of the entire cloud computing market, which itself is, say, a few hundred billion. But we're talking about 100+ billion dollar market that literally did not exist five or six years ago for a space.

So, I think it's going to continue to grow. I think we're going to continue to see lots of innovation, lots of models evolve and see a lot of iteration. But I think at the end of the day, we need sort of a first principle's mindset and thinking as it relates to the relationship between projects that create most of the value and the companies that are capturing some of that value and sort of – The first principle's thinking part of this is really critical, because if you sort of just look back at the traditional software tech company building approaches –

[00:50:33] JM: History doesn't give us much guidance.

[00:50:35] JJ: History gives some viable guidance across lots of different dimensions, but I would say if you just sort of –

[00:50:39] JM: We're in new territory. We're in very new territory.

[00:50:40] JJ: Yeah, new territory. I think if you can carry over a lot of the things and generalize, you end up with really bad, dangerous conclusions.

[00:50:46] JM: Right.

[SPONSOR MESSAGE]

[00:50:55] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:53:15] JM: And it is the strangeness, the strangeness and the newness of this business landscape that makes me excited about your conference, and happy to be a part of it.

[00:53:27] JJ: It's awesome to have you MC as well, by the way.

[00:53:29] JM: I'm looking forward to that. We didn't get to like a fifth of my question –

[00:53:35] JJ: I apologize for the really longwinded answers.

[00:53:35] JM: No. I think this is actually –

[00:53:38] JJ: If you want to go through the questions, I can try and force myself to give you 10-second answers.

[00:53:42] JM: No. Writing the questions were a process of getting my mind in the zone to talk to you. Just to wrap up, because I think we've really focused this entire conversation on – I mean, we actually did – What we're going to do this on is permissionless innovation. I think at this point you've defined what permissionless innovation is, or you defined some parts of it. I think permissionless innovation is a broader philosophy that we have time to entirely cover in this podcast. But just looking at your writing, like I kind of get it a just for where your head is that.

Just because we've focused on the interaction between commercial open source companies and cloud providers for so much of the show, I want to close by just talking that some concrete strategic alternatives to changing your licensing model. Because I have sympathy for these companies, like Redis Labs or whatever. You've raised a ton of money in venture capital. You

need to deliver venture skill returns to your investors. You've put so much effort into building this amazing in-memory object storage system and you get – Amazon just has ElasticCache, and they probably capture as many customers as you're able to, and that seems frustrating. Why is Amazon able to just deploy, basically, our same code base and without the effort and monetize? That can seem frustrating with company like Redis Labs. Then they say, "Okay, yeah. We're going to change our license."

My stance, and this is totally not having much insight into what's going on at Redis Labs or the engineers there, is that you're a software company. Figure out a new software product to build. But I want to understand from your point of view, if I'm Redis Labs and I'm threatened by this, what alternatives do I have to changing my license? What are the other products or services or business strategies that I can employ rather than changing my license?

[00:55:42] JJ: I'm going to go back to the perceived threat and the perceived existential dilemma these companies are in and that they view is, again, an non-problem.

[00:55:53] JM: Wow! So you think they're business models.

[00:55:54] JJ: And they are constructing –

[00:55:55] JM: The business model is not even a problem.

[00:55:57] JJ: No. No, I didn't say that, but I'll say what the perceived threat is. Business models always requiring constant innovation and evolution. That's an independent threat. What I think is the view that cloud providers are posing an existential threat to commercial open source vendors is categorically false based on faulty and highly flawed conclusions and perpetuated by, again, the conventional zero-sum thinking applied to a world that is fundamentally governed by positive-sum dynamics.

[00:56:31] JM: Wow! I see what you're saying. Now I get it.

[00:56:33] JJ: Now, I'll give you an alternative because you asked about what is an alternative approach. An alternative approach would be – And it actually is reflected in – I'll give you a few

companies that I think are really building amazing businesses that have the potential to be 5, 10 plus billion dollar companies, and they're very well on their way to doing that, and none of them have embraced any new kind of what I would say is discriminatory or non-compete kind of license model. In doing so, I think probably would be negative, but certainly they haven't even had to evaluate that up until now.

Again, let's just look back at the phenomena here. This whole trend is a year, maybe 18 months old. It's really actually maybe a-year-old, and the vast majority of the companies in the index that we've been looking at did not have meaningful businesses at all north of a few million in revenue five or six years ago, and now the space and the category in terms of aggregate market value of at least the top 40 companies, and there are actually a few hundred that are doing reasonably well, is north of $130 billion.

So, let's look at I think a positive way that you can construct and optimize on positive-sum collaboration with many constituents capturing value around your software. You can look at what HashiCorp has done. They have MIT-licensed projects. They collaborate very tightly with cloud providers. In some cases, they have partnerships and royalty engagements and co-marketing efforts and so on. But their projects are very much so fully open source. Anyone can contribute to them. Anyone can extend them and distribute them and run them and so on, and they've built a business and an open core model that sells proprietary enterprise products based on their open source projects to the global Fortune 2000. They've build a –I think last I checked, and I believe they're public about this. What is very fast approaching a multi-hundred million revenue business that actually wasn't really more than a few million in revenue just a few years ago.

Another really great company that is very well on their way to 100 plus million revenue business soon, and they're super transparent about that, is GitLab. So, GitLab has what they've called, I think in a very innovative way, a buyer-based open core model. So they've actually constructed enterprise versions and obviously proprietary products that are sold to different types of buyer personas in the enterprise.

[00:59:07] JM: GitLab for oil manufacturers.

[00:59:09] JJ: It's more like GitLab for VP, for CTO for director of engineering, buyer-based. So, persona-based. Industry-agnostics, so it's horizontal, but it's buyer-based. Then the core open source project is completely open source license. They have a great developer community and they're very transparent about where they draw the lines in the sand effectively in terms of the open source project and the closed proprietary extensions. So I think that's very clean and elegant and simple and it's very scalable. It doesn't introduce complexity in sort of mixed incentives and sort of complications.

So, I think there are a good number of examples. I just mentioned two, but there are a few more that I can give you. I didn't want to give you a 10-minute answer, but there are a number of examples of companies that rightly so appreciate that the software that is creating most of the value is decoupled from company, and that project and that open source software should be cared for, invested in, government and produced for any constituent interested in capturing value around it, and that includes a cloud provider. That includes competitors. That includes anyone. That is a really great paradigm.

Now, abstractly, if you look at that for me zero-sum sort of monopolistically capitalistic approach as supposed to positive-sum meritocratically capitalistic approach. It's really easy to sort of say, "Well, that's crazy. I'm not to be able to build a big business on that at all."

I would say as a response and, again, I mentioned this a few times, five or six years ago, there were less than four companies that were commercial open source software companies generating more than 100 million revenue on a run rate basis, and it is really mostly Red Hat's Hussein and I think maybe one other Linux vendor and Cloudera.

Now we have 40 these companies and they're all generating north of 100 million in revenue each. They're all growing really well. Some of them are not growing as well as others. Some of them have issues, like recently we saw Pivotal's public market issues and some of Hadoop ecosystem has some issues. Fundamentally, the broad class of companies are doing really well, and they operate under this meritocratically capitalistic structure, whether they sort of appreciated that or not, or maybe like stumbled into that reality over time. I think that the model works really well, and I think it's better approached for the world. It's something that creates

more value for the world at the end of the day, and it's a future that I certainly welcome over one that is governed by fully closed proprietary ecosystems and platforms.

It's not to say that today if you started a fundamentally, fully proprietary platform company or software company, you will not be able to be successful. I think we're going to live in a highly heterogeneous mixed reality world probably for decades, and that's a good thing. We need variety in the world, but I'm for an open future and one that is governed by these kind of commercial open source principles.

[01:01:59] JM: What I like about talking to you is many times I am confused in the middle of our conversation, but by the end of it, I see where you're coming from. It's a pleasure. I mentioned this to Pesam in an interview with him recently. I initially was kind of skeptical of the open source software capital. Not skeptical in terms of like I never had any doubt that you would have a great deal flow and that you'd be a great picker for investments, but the whole idea that open source software companies are really that different than normal software companies. I was like, "Eh! I don't think it's really that different. I think this it's just as you show some of your code to the public."

But, now, I am realizing like even just as this licensing example as one example of something that makes open source software companies, like this is a huge strategic inflection point for some of these companies. Changing your license, that could have catastrophic effects for these companies. Like you said, we don't know yet. We don't know how it will impact them, but I don't know. Anyway, all I'm trying to say is I'm a fan of your thinking, and it's interesting to talk to you, consider your branding and your vertical focus on OSS Capital validated in my eyes. Thanks for coming on the show.

[01:03:25] JJ: Thank you, Jeff.

[END OF INTERVIEW]

[01:03:30] JM: GoCD is a continuous delivery tool from ThoughtWorks. If you have heard about continuous delivery, but you don't know what it looks like in action, try the GoCD Test Drive at gocd.org/sedaily. GoCD's Test Drive will set up example pipelines for you to see how GoCD

manages your continuous delivery workflows. Visualize your deployment pipelines and understand which tests are passing in which tests are failing. Continuous delivery helps you release your software faster and more reliably. Check out GoCD by going to gocd.org/sedaily and try out GoCD to get continuous delivery for your next project.

[END]