EPISODE 709

[INTRODUCTION]

[00:00:00] JM: Open source software powers everything that we do on the internet. Google runs on Linux servers. Content sites are served by WordPress. Our data is queued in Kafka clusters and stored in MongoDB instances. The success of an open source project often leads the creator of that open source software to become wealthy by starting a business.

An open source project can be monetized through enterprise add-ons, or consultation, or simplified hosting. The creators of open source software know their domains so well that they are usually well-suited to operate one of these kinds of open source business models. Open source business model success stories include Elastic, which was based on the elastic search open source project; Cloudera, which is based on Hadoop; and Red Hat, which is based on an enterprise version of Linux.

The rise in usage of cloud providers has change the viability of some of these open source business models. Amazon Web Services can monetize almost any open source project more profitably than the creators of that project, and this is because AWS has established distribution channels with all of the people who are running their server instances on AWS.

If I already run my application on AWS and I'm looking for someone to provide me with a hosted version of a database, or a search engine, I will probably choose the hosted database that AWS provides, unless I have a good reason to choose a different provider.

The Commons Clause is a licensed condition that open source projects can use to protect their code from being profited from. Redis, which is an open source in-memory object storage system, recently added the Commons Clause to their license with the goal of improving the business of Redis Labs, a company built by the creators of the Redis project. If this sounds confusing already, don't worry. We’re going to explain it in this episode.

Kevin Wang joins the show to discuss everything about open source from business models, to security vulnerabilities, to licensing. Kevin is the CEO at FOSSA, a system for managing open source licensing and security. Kevin was involved in the creation of the Commons Clause and has written about in detail. He is also one of the foremost experts I've met on software licensing. I believe that FOSSA is the second platform he has started around open source licensing. So he is a dedicated expert in the field of software licensing and knows a whole lot about it. I hope you enjoy this episode.

[SPONSOR MESSAGE]

[00:02:59] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW]

[00:05:07] JM: Kevin Wang, you are the CEO at FOSSA. Welcome to Software Engineering Daily.

[00:05:11] KW: Thank you so much for having me.

[00:05:14] JM: Fosse stands for free open source software analysis. Explain what your technology does.

[00:05:20] KW: So we're an open source project, and basically we built a tool that you can plug into all your build processes, CI/CD pipelines, and then we'll help you track and manage all of the different open source dependencies that you use. It's great for the individual developer, but the magic really happens when you get to deploy that scale across hundreds of build pipelines across a bunch of different teams. We do that in a bunch of companies like Twitter, Zendesk, Docker, Docker, Oath, etc.

[00:05:46] JM: The first area that your open source software analysis software tool has focused on is compliance and license management. There's a lot of people out there who probably don't know why this would be an important concern. Why is compliance and license management of software an important problem?

[00:06:08] KW: Yeah. The most important products on top of our technology is compliance automation suite. Basically, what it does is it goes through all your dependencies, it scans every line of code for different licenses and then it will just do the job of collecting all of the hundreds of licenses that you might be using across your open source projects. Then what we'll do is we'll automate things like reporting and compliance documentation. We'll flag any violations we see. We can help you enforce these policies. So that's what the tooling does.

The reason why it's such a need is I think there're a couple of things that have really changed. The first piece is most software companies now in any product that they ship is using up to 80% to 90% third-party code. So if you're using an app today and you're running some code in production, that stuff is third-party IP. It's not written by your developers, and every single one of those is under these hundreds of different types of licenses. A lot of them are very unique. A lot of them have very unique types of interactions. So that kind of scale of legal complexity is just something that a lot of legal teams can't really handle anymore, and it's super important, because it's the majority of your product.

The second piece is that legal risk is just one of those things that escalates really, really quickly. You can have one very small kind of violation that can spiral out of control into something that really post a lot of business risk. While some issues, especially things like code quality issues can be contained to some codebase, legal issues are one of those things where if you just have one thing that's not fully crossed off, it could be a big problem.

[00:07:43] JM: I don't think I have heard of any cases of software companies getting sued because they are not adhering to the licenses of the open source software that they're using. How does this legal risk actually manifest? Do companies ever gets sued by open source projects?

[00:08:02] KW: Well, all these stuff happens all the time. One of the major issues is it’s something that usually isn’t super public. It usually isn't something that goes to court till way beforehand. Specifically, when license violations happen, partially it's from a violation of open source licenses. But the other part is a lot of the times, and this just plays into the narrative of how hard it is to control your code as a modern company, is that you might be using proprietary software and you don't even know it. Across most of the open source registries out there, like NPM, or GitHub, or etc., you'll find tons of code that’s just out there, but it’s on license, and it will just turn out to be proprietary code.

We’ve seen the most crazy instances, like we've seen some companies actually become these sort of like Kyparite trolls where they’ll intentionally publish packages that look like open source packages, but have some fun home mechanism and all it does is like par states or do something super, super trivial, and then they’ll go out and then sue any company that uses it.

Those types of things are super exceptional. They’re not very common. I would say that it's not likely that you're using something that belongs to like an open source licensing troll, but there is a lot of stuff out there and it's just very, very hard to keep track of what all of the different licenses you're using.

[00:09:19] JM: Yeah. I guess it reminds me, I guess, could be as a podcaster, it reminds me of this whole patent trolling around podcasting a while ago where there were podcasters whose entire business was threatened by one patent troll who happened to say, "Oh! I invented what the podcast is, and here's my patent around it, and threatened Adam Corolla's business and Mark Marren's business," and they had to spent a lot of time fighting it. It is unsurprising that companies would want to pay for a solution that just takes this risk off the table or at least makes it a very small risk.

[00:10:02] KW: Right. Again, our message here is not to spread fear, and that's one of the things that I kind of hate about the marketing especially around things like open source compliance solutions, is it seems like the traditional marketing says, "Hey, look at all the stuff they have to be afraid of, be afraid of open source," and I think our message, especially as an open source project itself, is don't be afraid of open source, and open source is amazing. It's something you have to use regardless, instead embrace it. Be good at using it and use tools that can automate and manage this kind of risk at scale instead of having your legal team manually audit this kind of stuff.

[00:10:39] JM: One thing you're implying here is that when you find a repository on GitHub or NPM that you want to use and you bring it into your software, there is a multiplicity of different licenses that that software that could potentially be under, there's LGPL and – I don't know all the acronyms, but maybe you could tell me about some of the ways that the licenses vary. Even though all of these projects may be open source, how are the licenses vary?

[00:11:14] KW: Right. So there's a couple of major categories of licenses. For all intent and purposes, I think that most people care about whether they just can use it and not think about it, or whether they have to really, really consider whether they can use this license or not. Inside of the open source world of, the two most important categories are permissive an copyleft.

Permissive licenses basically say, "Hey, you can pretty much use us for whatever purposes as long as you give some minor attribution or something like that. We're all good." The copyleft licenses usually say, “You have to use us, but you're obligated to make sure that this code stays open.” So they’ll have things inside of it like the GPL is very famous for saying, “Hey, if you use this GPL software, it has to be incorporated into other software that's also GPL and open source.”

So the copyleft feature is basically try to perpetuate the open sourciness of whatever it’s integrated into. For tons of reasons, this can be very problematic for businesses that want to protect their IP. But that's the general side on that category, and the permissive licenses are very open. Inside of the open source diverse, and when I say that, I don't mean open source license projects, but just things I GitHub and NPM and all these registries. You’ll also find a bunch of other really creative licensing schemes too. You’ll find a ton of source available licenses, which aren’t open source licenses. They’re license that preserves some of the features of open source, like having code availability, but will require payment or licensing if a certain threshold is met. You’ll find a bunch of other proprietary types of licensing schemes and also just tons of unlicensed code in general.

So there're just a lot of different kinds of licensing schemes. Many of them have different compatibility issues, and so a lot of permissive licenses are fully compatible with a lot of the copyleft or proprietary licenses. So those are all some examples of things to look out for.

[00:13:12] JM: There are people listening right now who are minutes away from turning off the podcast because they're thinking, “This is not interesting to me at all. This is like licensing management. Why would I care?” But one thing that I find interesting about it is that licenses can – If you have a creative license, you might be able to build a business model around an open source project, whereas without intelligent licensing, you would not be able to build a business around it.

So people who are listening who are entrepreneurial should be paying attention, because the idea of an open source business model has been appealing for a long time, but it's elusive in many ways both for contemporary reasons and for fundamental reasons of open source. Tell me about the relationship between licenses and business model viability and open source.

[00:14:12] KW: Yeah. I totally agree with you, right? I don't spend my entire day reading through licenses and getting excited about it, but the really interesting thing about licenses and the reason why they're so hard to read and dense is they're the building blocks for human intent. So they're effectively the medium that carries your ability to commercialize something like an open source project or build the business, or make sure that you get to import this new feature without having to build it yourself. They're really, really important building blocks and it's important understand them.

Most of the open source licenses out there, they don't really have any way of capturing some sort of business model, but there's tons of licenses out there that are really popping up now that are on the topic of open source commercialization. If you want to create an open source project, you hit it big and you want to be able to stay in that open source project by either creating some sort of proprietary offering or proprietary version of your open source project, then you would be looking for a new licensing scheme to say, "Hey, for these sorts of use cases, it's commercial, and we're going to use that to fund the development on top of open source.

[00:15:18] JM: Right. I think I want to go ahead and dive into this, and I want to come back to your business and some other elements of open source a little bit later on. But this question of the commons clause is something that I know people really want to hear about, and I want to go into as much depth as we need to here.

So there are issues not only in how open source software is consumed, which is what your business, FOSSA, is about. How are you consuming the open source software? Are you consuming it in accordance with a license? But the production of open source software, how open source projects can capitalize on those pieces of software that they're building, especially relative to cloud providers. So there is this issue that's epitomized by the Redis Labs question, where I start an open source project. Maybe it's an open source database, or an open source in-memory object storage system. It's doing great. There's a community behind it," and then a cloud provider stands up a servicized version of it, for example, with Redis. Redis Labs is a company that was started around the Redis object storage system that's open source, and there's also Amazon Redis. So Amazon Redis has been able to capture much more of the value of the Redis project than Redis Labs. Explain more about what happened with Redis and Redis Labs and Amazon Redis.

[00:16:46] KW: Right. So before we dive into this, I want to preface it with a couple of things. First, I think the discussion that has popped up around things like the commons clause has conflated a bunch of different kinds of issues. So I want to draw a couple of lines. The first thing is we should not confuse the commons clause with open source. The commons clause is definitely not an open source license, and for some background, it's effectively a license addendum that you can apply on existing open source licensing scheme to turn it into a proprietary license. Specifically, it’s like a source available license, which means that you can turn to in a project where you still get a lot of the features of an open source license. You get to see the code and modify it, distribute it, but you just can't sell exactly the same thing. But for all intents and purposes, this is not open source.

One of the things that happens is, first, whenever you start open source project, and I think this is completely clear across the board, you're pretty much making a commitment saying, “Hey, this thing is going to be open. It's going to be free. People can do what they want from it,” and you’ve made that commitment to make sure that you don't put any restrictions on top of it, and you can't really take that back because if you ever try to change the license on top of it, people can fork the older versions. There's really no – It's very, very hard to reverse that action.

When something like the commons clause comes out, it's not really going to go through and turn open source projects to something like close source projects. In the Redis example specifically, Redis develops a very popular open source database. The kind of action here is not to like figure out, “All right. How do we commercialize and take the most advantage of an existing project?” But Redis, if you recognize how it’s developed, I think like 90% of the committers work for Redis Labs. Most of the development is funded by this one specific source. So this source has one really important way of making money, which is selling hosted and manage Redis. If you take that away, then that really threatens the future of the open source project.

So what Redis did was Redis took some of the enterprise modules, which are part of their commercial offering. They're not part of the core open source project. It's not going to affect any of the developers, and say, “Hey, in the future for these enterprise modules only, for this commercial part only, this is now officially a commercial project.”

The important part of that is in order to retain that kind of ability to commercialize, you're actually sustaining the ability for the open source project to succeed. So the move here was really to protect the thing that was funding the existing open source project from extremely powerful and large businesses, like the Amazon, the Googles of the world, from being able to user their channels and completely bully everybody else out of the market.

[00:19:45] JM: Right. One thing to point out here is that while Amazon may be capturing the most value from Redis through Amazon Redis, they do not contribute code back to Redis. Is that correct?

[00:20:01] KW: In most cases, not much, and that's okay, because it's under a license that basically says, "Hey, we've given you permission to do whatever you want with it." So they don't do that, and that's not something I think to sort of necessarily be frowned upon, because Redis is explicitly given permission that this is an okay thing to do. But I think that also means that it's a totally okay thing for Redis to do to take some of their commercial code and explicitly say, "Hey, this is now commercial code moving forward."

[00:20:35] JM: What were the modules that Redis, the project, the open source project, decided to put under commons clause?

[00:20:44] KW: There are a couple of modules that were exclusively created by Redis Labs. I think they're Redis Search, Redis Graph, ReJSON, Redis ML, ReBloom, but they're effectively these add-ons on top of Redis that go into their enterprise offering. So it's nothing about – It's nothing something inside of the core open source project that's Redis. Redis is open source. I think Redis made it very clear that's always going to stay under a very permissive and open source license. So the only thing that's changed is future versions of their enterprise modules.

[00:21:19] JM: Okay. Just to be clear, what does – Putting those modules under the commons clause, what does that change?

[00:21:28] KW: The commons causes is basically – It's very short. You can go to commonsclause.com to read what it's about, but it pretty much says that you can do whatever you want under the existing license grant, except for essentially the same thing.

I believe those modules were previously license under Apache II, which is a very permissive license. It’s one of those licenses that says you can mostly do what you want. The commons clause is just added on top of it, saying you can still use it under the terms of that license, except you just can't sell substantially the same thing. What that means is that you can't just package it up into a cloud offering and sell exactly the same thing, or you can't just take it, toss in a CD-ROM, that I don’t think anybody would ever do that anymore, but ship it out to somebody, but you just can’t sell just that software on its own. You can still provide can still provide services on top of it, you can still integrate it into commercial applications if it’s like a module. So if you're using like ReJSON for some JSON data parsing and you want to bring into some application, you can still totally do that and sell that whole application, but you just can't sell exactly the same thing.

[SPONSOR MESSAGE]

[00:22:46] JM: Managed cloud services save developers time and effort. Why would you build your own logging platform, or CMS, or authentication service yourself when a managed tool or API can solve the problem for you? But how do you find the right services to integrate? How do you learn to stich them together? How do you manage credentials within your teams or your products?

Manifold makes your life easier by providing a single workflow to organize your services, connect your integrations and share them with your team. You can discover the best services for your projects in the manifold marketplace or bring your own and manage them all in one dashboard. With services covering authentication, messaging, monitoring, CMS and more, Manifold will keep you on the cutting-edge so you can focus on building your project rather than focusing on problems that have already been solved. I’m a fan of Manifold because it pushes the developer to a higher level of abstraction, which I think can be really productive for allowing you to build and leverage your creativity faster.

Once you have the services that you need, you can delivery your configuration to any environment, you can deploy on any cloud, and Manifold is completely free to use. If you head over to manifold.co/sedaily, you will get a coupon code for $10, which you can use to try out any service on the Manifold marketplace.

Thanks to Manifold for being a sponsor of Software Engineering Daily, and check out manifold.co/sedaily. Get your $10 credit, shop around, look for cool services that you can use in your next product, or project. There is a lot of stuff there, and $10 can take you a long way to trying a lot of different services. Go to manifold.co/sedaily and shop around for tools to be creative.

Thanks again to Manifold.

[INTERVIEW CONTINUED]

[00:25:00] JM: Now I thought that this whole debate around the commons clause was interesting. I thought it was a step forward for open source innovation, because it doesn't restrict open source in any way. It expands the models of sustainable open source. So that was my value judgment on this whole evolution in open source. But there were a few – At least a few very loud voices who were decrying the creation of the commons clause.

[00:25:33] KW: There were a lot.

[00:25:35] JM: Were there a lot? I mean, relative to the amount of developers, I think most developers are like, "Okay. I don't care about this at all." Okay, there were some loud voices.

[00:25:45] KW: Okay. There are some very loud voices, very loud voices. Here's what I have to say about that. I think whenever you see something like this pop up, and for those listening that are not caught up with the news, there is a lot of controversy when the commons clause first launched and a lot of people saying, "Hey, this is going to be the death open source."

For a variety of reasons, it's obviously not true. We explained that too and addressed it on the website, but I think the first place is to just look at who's saying what to understand where this –

What type of opinion to really develop around something like this. I agree with you, I think the commons causes is another option that people can choose to be able to fit their needs. Frankly, it's an option that's made as an alternative to going complete proprietary, because there are sometimes, when an open source project just can't sustain itself with its existing license model, so it has to – It's like, "All right, we have to make a licensing change. We have to go proprietary," and this is just something that's a bit more in the middle that's says, "Hey, you can still have an open source code base. You can clearly state, "Hey, under these conditions, it's a very narrow thing that promotes this kind of commercial use."

Pragmatically, it's something that's in the middle, and I think we've seen responses from two different camps. One camp is people who sort of are there to steward and defend the term of open source and defend what it means to be an open source license, and I think that's very, very true. In that case, I can understand why somebody would get really upset about this, because they see something like this. It doesn't exactly fit into an open source license definition, and that can be pretty confusing for people.

On the flipside, most of those voices have not come from actual software developers themselves. But what the software developers seemed to be saying is saying, "You know what? It's been so long since we had a new license that can allow us to build a business model around the things that we're doing." We hear frustrations like it's so hard to create an open source project, make it successful and get paid for it and sustain it. We've seen a lot of positive outpour and just this general sentiment that the existing licenses out there don't really fit our needs on what we need as software developers. So something like this is really, really welcome.

Since then, we've seen a bunch of different companies like [inaudible 00:28:06] j Dgraph, etc., adopt the commons clause. The other day on Twitter I saw Michael DeHaan, who is the creator of Ansible, adopt the commons clause first new project, vesping.io. I've also seen MongoDB come out with the SSPL, which a change in the GPL. The whole idea is I understand how one license proliferation is really confusing and it's really bad. But on the flipside, I think people need to recognize that the last really popular license is written about 10 years ago. In the span of 10 years, we've have the cloud revolution, we've had containerization, we've had all these fundamental shifts in technology. Open source is completely different today than it was 10 years ago. It’s about time that there're a couple of new licenses to address all of these changes and serve developers in the modern era.

[00:28:56] JM: MongoDB is another interesting data point. Can you explain what happened there?

[00:29:04] KW: Yeah. I think MongoDB came out with its SSPL –I forgot exactly what it stands for. It’s a new open source license that they came out with. What it seems to be is it seems to be something like a fork of the AGPL. So it seems to be targeting kind of like a similar goal, I'd say as the commons clause. But in something that tries to fit the open source definition a little bit closer. If I'm aware, basically, MongoDB, when they changed this license, they also submitted it for approval against the open source initiative, which are the folks that kind of guard the term of what open source means, define it and approve different licenses so it is actual open source or non-open source.

I don't think it's gone through the approval process yet. So as of now, it's in the non-open source category, but it seems to me to be a bit more into the category of we’re trying to be an open source license and here's a way that we’re going to do it.

[00:30:08] JM: Yeah, I didn't exactly understand the terms of the license, but the motivation seemed the same. There were some cloud providers that were productizing MongoDB in a way that was cannibalizing some of MongoDB, the company's business, I guess.

[00:30:25] KW: Here’s a bit of history I think to fill some people in. A long time ago, when we were – And this is why software licensing is like really, really crazy and hard. A long time ago when we were just reading software, we'd like throw them on CD-ROMs or floppy disks or you like download them directly from a server. The way that you can control how the stuff was getting used was really clear. It's basically distribution.

If you're getting a CD-ROM, if you're like downloading this thing, then it's clear how you can control that point. A lot of licenses were written around this idea of distribution, but then everybody moved online and then now you're using websites in your browser and you're not like downloading all of the software anymore.

So here comes this license called AGPL, which tries to get around it and it basically changes the term distribution to something like conveying, which basically means the scope of licenses no longer on downloading the thing. It's on like if you're even using it through a web browser or something. Then now with cloud providers, there's sort of like a new way that they're using software, which is take a service like Redis or Mongo and put it into a managed service, and that doesn't really fit the idea of conveyance that the AGPL is drafted under, or the idea of distribution which most licenses were drafted under. It's kind of this like whack a mole of different issues, where it's like, "All right. Here's a new way of writing software. All right, let's run and chase it and then whack that mole down with a new license."

I think that's one of things that I understand is sort of scary from the open source side, is, "Oh my God! There's going to be so many different licenses. It's going to be so hard to keep track of what's going on." The flipside is if you don't have any experimentation over how new licenses go, you're going to actually increase the chances that you're going to have this sort of like whack a mole license structure. I think it's better to sort of try to come up with blanket licensing schemes that can cover a bunch of different sorts of use cases.

[00:32:19] JM: Now the cynical person that's listening is going to say, "Okay, I'm hearing this from Kevin Wang, the same person that's trying to sell me software that allows me to manage all of my licenses in an automated fashion."

[00:32:33] KW: Yeah, totally. That's a pretty good point, honestly. Somebody told me that on Twitter through this like whole debacle and I was like – Or I think it was in Hacker News or something. I was like, "Oh, man! I guess that is sort of like a weird thing."

So here's the thing. What I would say to that is if the intention here was for us to grow our business by making licensing more complicated, then there are probably easier ways we could do it. We're a growing startup, so we can hire more salespeople. We can focus on better content marketing, improved features, etc. So there are like much, much easier ways I think for us to grow our business than focus on stuff like that.

I think the real history of this is we're a pretty popular license management tool. I think we're the most popular license management tool around open source projects, around developers. We don't put up content that says, "Be afraid of open source licenses." We try to be very open source towards sales. We have open source projects that we maintain under permissive and copyleft licenses, and we generally are pretty transparent in how we do our business.

How we got involved in this is just we're a part of a lot of different licensing discussions and a lot of people tend to come to us when they need help with different licenses, when they want to do something open source related but they don't internal expertise around it, or they want to figure out what's the best way to come up with, say, some new licensing scheme or pick a license for their project. So we just end up in the middle of a lot of these stuff.

[00:34:11] JM: In your defense, I was looking at your background, I think you've been passionate about – As far as I can tell, you've been passionate about software licensing since you were like a young teenager. Is that accurate? You built some indexing software over all these different licenses and stuff. It seems like you are genuinely passionate about this, and to be making things unnecessarily complex and overly licensed would be – You'd be selling your passion in short in favor of making a quick buck, which doesn't seem like something you would do.

[00:34:50] KW: Yeah. In high school, I think the first thing I've built around this stuff was a site called TLDRLegal, which summarized software licenses like open source licenses into these simple summaries that are plain English, human readable, but you can cannot must do under the license. Since then, millions of developers have used that thing. It's inspired GitHub's license user and the GitHub license pages as well.

Yeah, I what I saw is I was a kid that always into open source. The open source community was how I learned how to code. One of first jobs I ever had, I worked at Cloudera working on the Hadoop the ecosystem. The main dangers that I saw against open source were things like bureaucracy and things like FUD and people just misunderstanding things like licenses and being really scared of them.

I feel like our mission is to kind of take the fear out of the stuff, because you know what? I am not concerned at all about the ability for lawyers to be afraid of new licenses inside of companies and then make sure that companies act really costly and slow things down. So I'm not concerned about that all. I think the real answer to all these stuff is just create license schemes that are simple and pragmatic and then just build tools that make it really easy for people to figure out what's going on and do it from a developer’s perspective, because frankly it's developers that are on the front lines of all these stuff.

Yeah. I mean, I think that's the way that we want to try to solve a problem. Again, we don't really go out there fear mongering on licenses. I think there's been more than enough of that and I think that's just a net negative for the entire industry. I think our messages is, “Hey, this stuff doesn't have to be scary. Look, you can pick up this tool, use it in the next couple of minutes and get a whole compliance process running at your company that probably beats a lot of the manual legal review processes that people have previously invested millions of dollars in.”

[00:36:42] JM: I think another sentiment that I saw in some of the people that were opposed to the comments clause coming into existence was why does Redis Labs need to do this? Could they'd figure out some other creative way to monetize their business? You look at a company like Elastic. Elastic is going to IPO. They've got software that's built on top of open source and they've managed to make it work to the extent of an IPO even though AWS has their hosted version of Elasticsearch. What's the difference between Elastic and Redis Labs?

[00:37:21] KW: The context here is that we operate in business, and I don't think the comments clause is a perfectly drafted license. Frankly, I think most open source licenses out there are pretty poorly drafted, like the MIT license even, the most popular open source license out there, super short. Doesn't really say anything about a lot of important things that lawyers would think are important, like patents. But, frankly, in the world of business, things are constantly moving. For a company like Redis where there are extremely valuable releases that are launching ever quarter or so, getting something done and shipping something is more important than I think theory crafting the absolute most perfect solution that’s sustainable over time.

I think whenever you create something like a new license, it's definitely important to be very, very careful of it, but there's just a whole intent on what it’s supposed to do. You want to maybe like trigger discussion with people that are using it under certain ways that you feel like are uncool. Even if you were to spend like a whole year coming up with this super perfect 45 page license, well then the risk that you have is, first, there's always going to be people that misinterpret it or don't think about it in the right way or find a bunch of problems with it. The bigger risk is if it's this long, perfect license or thing that has been sort of like overly theory crafted, people are just probably not going to read it and label it as like, “All right, this is too long. TLDR,” and then just not use it all.

I think from what I've seen so far is that some of the most successful open source licensing schemes are really not the most pure from a licensing perspective where it’s just like simple and easy for developers to get the gist of what the intent is an easy to sort of distribute at scale. I think the reality what happens at Redis is Redis is a business and they’re moving fast, and they're growing fast, and they’re doing incredible things. They just needed to get something done and get something at the door.

[00:39:23] JM: The comments clause is an example of a solution to one market failure within open source, or what I would say is a market failure. You've got Redis Labs developing software, and AWS capturing the vast majority of it. Another market failure that comes to mind that we covered recently was Babel, for example, and Henry Zhu who works on Babel all the time and every major – Well, I don’t mean every major website, lots of major websites use Babel, but it’s just an open source project maintained mostly by this guy, and he doesn't really have a way to get paid for it.

Now, I think he's found success in a Patreon sort of model getting paid just by people subscribing to charitably donate to him continuing to work on it. It’s not entirely charitable. It's also out of their own self-interest. A company like Facebook wants to keep having access to Babel. With that as another example, what are the other market failures that exist in the world of open source?

[00:40:30] KW: I would push back on that. I don't actually know if that actually is a market failure, because I think we live in a capitalistic society, right? If you create a lot of value and you can capture a lot of value, you can build a really, really great business. I would actually argue that major, major open source projects that create value on the scale, things like billions, extremely, extremely important infrastructure have no problem getting funded. It comes in a couple ways. One is you can create a really successful business.

In 2018, we've had so many incredible outcomes, like Elastic and etc., that are proven that if you create a really valuable open source project, there are tons of ways to create extremely valuable businesses around them. So that's one market success.

The other side is if you create foundational open source projects that are really important, you can actually raise a lot of money without just having to create a business. There are foundation models where you can get support. You can go to businesses. There are probably just markets of service economies that can develop around it, and usually one of those services businesses would actually step up and try to fund the development of that project itself.

So there aren't real market failures. I think what we're seeing though, we're seeing some efficiencies, right? They're on the scale of small companies and small developers. So Babel is obviously really, really popular and pretty important project in the JavaScript ecosystem. But if you're comparing the amount of value that was created through that versus something like Linux or Kuberentes, it's like a completely different zone.

To make something like Babel successful, you just need to fund one or two developers. I think that's one of the limits of the kind of Patreon-esque model, is there might not be enough value that you can capture around something like that that actually warrants some sort of scalable market function. I would actually say that a couple of developers, even something like OpenSSL where you have a few projects that are really important that are a bit ignored, are in the grand scheme of things just cracks in the whole market system that has been generally very, very successful. W're still filling them, we're finding them. The Patreon type model and those sorts of things I expect will be relatively small compared to the size of businesses that are created around open source, but will do a great job in filling in the cracks and will be really important to ensuring like security of our future when we're building everything on third-party code.

[00:43:00] JM: I guess you could say if you have somebody like a Henry Zhu who can keep Babel going, or you have him and one other developer and they get whatever, 220K, in Patreon donations per year, you could argue that's a market success rather than market failure.

[00:43:20] KW: Absolutely. Again, here's one of the things that I want to bring up about how successful this market has really been, is open source is so new. Think about like everybody thinks about the open source community is centralizing around like GitHub nowadays. Remember, 10 years ago, GitHub wasn't even remotely close to what people think about it today.

There's a period time where GitHub was like people were saying, "Oh, yeah. There's that site where you can use Git. No, I'm going to SVN, because Git sucks," and that wasn't that far long ago. In that period of time you have these gigantic economies of scale, these billion dollar companies that have just turned out like hot cakes. These relatively large scale, like Patreon type funding models that would've taken so long to really develop and people generally understand the value of open source now. All of that stuff has happened so fast, so much faster than any other industry that have ever seen history.

This market has been really, really successful. Whenever anything grows and there's any fundamental change in technology, sure, it will have a couple of cracks and you can nitpick on the edges of, "Oh! This person's not as well-served as this other person," but that's the beauty of capitalism, right? When you see a need somebody can create something around and fill it, and right now we see a lot of projects that do just that and they seem to be going really well.

[SPONSOR MESSAGE]

[00:44:48] 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]

[00:46:37] JM: One example that comes to mind, and I don't know if you have any thoughts on this, but you mentioned Linux and Kubernetes and something that I feel falls somewhere in between those two on the timeline is Docker, and Docker, Docker the company, I think – I don't mean to disparage them at all, but I think they're struggling now. I don't know to what degree they're struggling. I think I saw them on this YC company list recently still. So I think they still have a "billion dollar valuation". Relative to how much value the invention of Docker created, it seems like Docker the company has not been able to capture much of that value.

[00:47:17] KW: Yeah. I mean, look, did Docker have the absolute ideal execution? No. I don't think anybody would say, "Yes, they did everything perfectly." But at the same time, think for a second about what criticizing somebody like Docker really means, because they've created a billion-dollar business and they were able to miss out on so many different types of opportunities, or like Kubernetes and things like that or like people say miss out, but they were able to do so little compared to what they could've done and they still created a billion-dollar business, and that's mind-boggling to me, right? That is something pretty, pretty incredible.

I don't really like it when people say, "Oh, Docker failed because they don't own Kubernetes when they should have and they didn't. They're not a hundred billion dollar company. They're only billion-dollar company." That term only a billion-dollar company is pretty fantastic to me. It's pretty incredible, because billion-dollar company is amazing, right? That's an insane crazy outcome. They're created, they're produced something that even if they didn't capture 100% the value of it, they've changed an industry, they've changed technology, they've – Actually, created hundreds and thousands of different jobs around containerization. I think they've done a great job. People have criticisms in their execution, but they've done a fantastic job overall.

[00:48:43] JM: That's the thing, is I agree with your assessment there. What concerns me is I'm not – I guess we'll find out, but I don't know if that billion-dollar evaluation is still there, because now you kind of look at the company and I don't know how you sustain a billion-dollar, but I don't know what their service contracts are or what not. So maybe they're doing just fine. I really hope they are, because they contributed a ton.

If they were to not be fine, like they had to raise money, a down round or something and then kind of got sold or something – I don't know. It would just be like – What would be kind of too bad, because that would be an example of the cloud provider sort of being able to eclipse and engulf an open source company but –

[00:49:32] KW: Right. Look, it's never going to be perfect, and I can't speculate about how Docker is doing internally, but there is this kind of narrative where if you create an open source project that's so valuable, you can do so little compared to what you truly see retrospectively 20/20. They can still create something that hits a billion-dollar value. That's amazing, like it's, "All right. What if they actually did win it big and beat Kubernetes or what if they did cerate the most popular containerization servicer, or what if they did do XYZ?" Those are all things that we can speculate on. I think, generally, a lot of people agree that they didn't do as much as they could've. But even with that, the tailwinds of just that open source project alone is incredible.

I will say though, the one real limit though is as soon as you create an open source project that scale, you're committed to having to maintain it. That open source project takes a lot of attention. This is something that we saw at Cloudera too. We just had so many developers that are just on staff. It's full-time contributors to Hadoop. We controlled Hadoop, but we didn't – We haven't known it, right? Other people at Hortonworks had come around and sell Hadoop offerings and do things like that too.

So, sure, there's risk associated with that, and I think one of the things that people don't appreciate is that when you create an open source project of that scale, so much of your time and attention as a company is sucked into making sure that open source project is successful, that this commercialization effort, a lot of the times is secondary. That's what I think companies need to get more credit for, is that these companies aren't people, like executives, rubbing their hands together and tapping like, "Aha! I'm going to figure out how to make money off all of these open source now." The people that are super dedicated to longevity of the open source projects, and I think Docker is just an example of people that are just so passionate about open source that may be the commercialization self was something that was secondary in their mind for a little bit too long while there are really focused on making sure the community was sustainable.

[00:51:33] JM: Yeah, definitely. Good points all around. Let's shift the conversation a little bit. I've been think a bit about the whole centralization around these things like NPM and GitHub, and I wonder if do ou have any opinions on the dangers of everybody just being blindly pulling NPM packages and cloning GitHub repos. Is there anything fundamentally dangerous about that?

[00:52:03] KW: Well, here's one thing that it really relies on, is it gives you the single point of failure, and could be either really good or really bad. It can be really good, because you have the team maintaining that is really great at security, really great at best practices, really great implementing the stuff, then you just have a really secure ecosystem. If they're bad at doing that, then you have a really insecure ecosystem.

So it depends on the implementation. From our experience, like we built tools that integrate with 20, 30 different package managers and registries and languages. So we've pretty much seen it all and how people do dependency management. I will say that there are definitely some ecosystems that do it really well and there are definitely some ecosystems where we question some of the decisions.

So I'll give you an example, and this kind of ties into a sort of like the fun of building code analysis tools for this stuff. The best ecosystem I think that I've seen by far is the Rust ecosystem. It was written by the guy who the – the patch manager that was written by the guy who previously did Ruby Gems, they have this philosophy where it's like publish less packages, but make sure they're super high quality, have a very hard constraints to make sure that breaking changes or things that are built into the versioning scheme of the registry itself, there's a lot of tooling that operates on top the language to make sure that there are very few bugs.

Now, inside of Rust, you have one of the most high quality open source ecosystems out there, right? But on the flipside, when you use a tool like NPM, people I think all recognize that the JavaScript ecosystem has been one of the most incredible and powerful ecosystems out there. When you're using stuff in NPM, you don't really know if it's good or bad. You don't really know who wrote it. You don't know if it's something that might have a ton of bugs or might have security vulnerabilities or licensing issues inside of it.

One of the things that makes it hard is some of the design decisions and implementation of tools like that in the beginning don't support a lot of features that, today, when we look back, are really, really important. One of the most important features of that is something called effectively either like repeatable builds or build determinism, which basically means if you have the same project with the same configuration and you build it in one machine and then you take that project and you put into another machine, you're going to get the exact same list of open source modules that are pulled in, the same versions, same modules and everything, and that's really not the case with most open source ecosystems today. Most open source ecosystems, if you want to build at 9 AM, or 9 PM, or have different kinds of build configurations or network conditions, it can completely change list of open source projects that you bring it.

What that really means is that if you're a company and are using a lot of open source projects, you really don't have control over what you're using. You don't really know if you can do nothing to your code and tomorrow you could have a critical security vulnerability, or a critical licensing issue, or some sort of code quality bug that can just pop in and affect your entire application. I think that's a scary thing to me, and I see the people moving into trends where they're trying to implement more of these deterministic features into the package managers, but we're still a pretty long way from there. That's why I think people are investing so much in scanning for the sorts of things and bring it as part of the pipeline.

[00:55:25] JM: Right. That's the second product that your company is coming out with. So like you've got continuous integration tools for compliance, but then you've also got static analysis tools for security risks, for mitigating security risks that are associated with open source packages. So while you're checking for open source compliance with licenses, you can also check to make sure that all the packages that you're using are up to date.

[00:55:56] KW: Yeah. Yeah, exactly. Basically, the kind of logic goes like this; if most your code is open source and third-party code, that means most of the security vulnerabilities are going to be in your application is now coming from these places too. So you need a new strategy for managing it. Previously, a lot of the investment towards product security has been around training developers to write secure code and making sure that your code that you write is a-okay, has the right kind of logic doing stack analysis on top of that stuff.

Now, the main area of risk is actually making sure that you use components that other people have wrote that are secure, and that's because most of the code and business logic that runs from your application isn't your own developers anymore. So people are shifting their investments to look towards open source and third-party things, and the type of analysis that you need is no longer just stack analysis of your code. You need dynamic analysis about how your build behaves.

Again, because your build can change from like 9 AM to 9 PM with no changes on your part, this is something you kind of have to run dynamically inside of your continuous integration or build process to really get an accurate picture of what's going on.

[00:57:03] JM: What's the vision for the company beyond static analysis and compliance risk assessment? Where do you think it'll be in like 5 or 10 years?

[00:57:12] KW: Yeah. I mean, our ambition is really to be the platform for using and managing open source at scale. So the reality is open source is here to stay and it's only going to get increasingly more relevant. So it's sort of a key part of your strategy, because I think all companies now are realizing that open source is this critical supplier. For them to the be successful and being a technology company, they have to know what they're using. They have to know how to control it, manage it at scale automatically throughout their entire development process.

So there're a lot of different prompts inside of it. We're starting with compliance. We've launched a security vulnerability offering. We're working on this code quality offering as well to help people make sure they're using modules of high quality. We have a couple of other things that we're putting together. But ultimately we're trying to build this unified platform for being successful with being open source. It turns out, if you're good at open source, it turns out you're probably good at being a technology company.

[00:58:08] JM: Agreed. Yeah, you could be like a white listed GitHub or I guess you could just keep building like random tools that people need and they'll keep paying you for them. That works too.

Ultimately, there's sort of two pieces. One, you have to be a part of the developer workflow, because they're the ones that are on the frontlines using this stuff all the time. But the business value really comes from the lawyers having dashboards to see what's going on and being able to get their reports. The business people seeing how quickly their software development process is moving based off what they're using and understanding how to make strategic investments in different ecosystems. It's from the security team being able to keep track of all the vulnerabilities that are coming in through their process.

This is not something that just spans like one developer. This is something that's an organizational why construction across a large enterprise with tons of different webs everywhere. I think there is no shortage of value we can provide. I think there're easily many billion-dollar companies that are going to turn out of solving this problem well. I think we've done a really great job so far with compliance and we're excited to see what's next.

[00:59:16] JM: All right. Just to close off, I want to ask you; do you have any other views about open source software, open source governance, things that might surprise people?

[00:59:28] KW: That's a good question. I mean, overall, I'm just get out really excited about open source itself. A lot of my time, just sort of for free time recently has been thinking about open source commercialization. One of the things that I'm surprised is still a surprising fact, but isn't just well understood enough, is that open source isn't free. Open source is very, very expensive to build and maintain. Software engineers right now are one of the best compensated jobs in American, and you need many of them, the top tier ones, to really create successful open source infrastructure.

I think one of the most surprising facts is something that we talked about a bit earlier, which is how new and fast changing this stuff really, really is. It’s like open source ecosystems are almost seasonal at this point. There's a completely new way to write JavaScript every three or four months, right? This little change is kind of unprecedented and we’re working in an industry where pretty much all of the building blocks to make ourselves succeed and make the next generation of products are almost free and open and easy to use. That kind of funding needs to come from somewhere, and that's something that’s just constantly changing and super accessible.

So when we think about open source commercialization, when we think about these topics and think about what it means for the future, I think we need to pause for a second and just be very grateful on what kind of industry that we work in, how incredible it is that if I needed to go and build this incredible UI or find some sort of way to store my data superficially, I can pull something off from across the internet that has had tens of millions of dollars invested into the development of it and just use it right away and get up and running. That's just completely incredible.

[01:01:13] JM: Well. I share your gratitude. We are attendees of the same church of open source software. Kevin, this has been really great talking. I really appreciate you coming on the show.

[01:01:22] KW: Great! Thank you so much, Jeff. I appreciate it too.

[END OF INTERVIEW]

[01:01:27] JM: We are running an experiment to find out if Software Engineering Dailydaily listeners are above average engineers. At triplebyte.com/sedaily, you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast. I will admit that though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself.

But if you want to check out that quiz yourself, you can help us gather data and take that quiz at triplebyte.com/sedaily. We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz course.

If you're looking for a job, Triplebyte is a great place to start your search. It fast tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself, and I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bits.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[END]