EPISODE 1337

[INTRODUCTION]

[00:00:00] KP: By most accounts, the first databases came online in the 1960s. This class of software has continued to evolve alongside the technology it runs on and the applications it supports. In the early days, databases were typically closed source commercial products. Today, many databases run in the cloud on distributed systems. Increasingly, the leading tools are open source, yet frequently supported by a related commercial entity offering managed services and white glove support.

In this episode, we interviewed Jonathan Ellis, CTO of DataStax, and Spencer Kimball, CEO of Cockroach Labs, about the current state of distributed databases, and the open source ecosystem.

[INTERVIEW]

[00:00:49] KP: Well, Spencer and Jonathan, both of you, welcome to Software Engineering Daily.

[00:00:54] JE: Thanks, Kyle. It's great to be back.

[00:00:57] SK: Thank you, Kyle. I'm pleased to join.

[00:01:00] KP: So, I hope listeners will go back and check out your previous appearances and learn a little bit more about your background and some of the things you've worked on. But for those who haven't, or haven't done that yet, maybe we can do a quick deep dive on your companies and various projects, perhaps starting with you, Spencer, tell us a little bit about Cockroach Labs.

[00:01:18] SK: Cockroach LABS is a relational SQL based database system. It's really aiming at that product category. Let's say the Oracle is probably the exemplar of the product category, but really reimagining what a relational database looks like when we have a global public cloud.

So, that I think really is embracing two concepts of scale. One is extremely large data sets, where you really want a relational database. And the other is geographic scale. So, how do you really make the database work well for customers, no matter where they are around the world. We're about almost seven years old now, based in New York, we're about 300 employees, and we're growing quickly.

[00:02:01] KP: Very cool. Jonathan, how about you? Give us the high level on DataStax.

[00:02:08] JE: Yeah, I started DataStax 11 years ago to commercialize Apache Cassandra, which is SQL database that is focused on availability at scale. We've had Cassandra clusters, go through Hurricane Sandy and never lose neither data nor availability. Because Cassandra is designed to run across regions, and accept both reads and writes anywhere in the world. So, for the past year or so, I've been working on a complimentary project, which is Apache Pulsar, and how do we tie that in to Cassandra to provide not just the system of record, but the messaging between services at scale.

[00:03:02] KP: When we move databases to the cloud, we get a lot of benefits, certain redundancy, distributed, things like that, I guess. What are some of the drawbacks? So why is maybe a single simple tenant, one database instance even if it's impactable, desirable, are there any challenges to moving to the cloud approach for databases?

[00:03:23] JE: know, I think the more you care about performance, then the more you need to understand how your system works. Most people are coming from that single system background. And so, in some ways, a Cassandra or a Cockroach isn't actually more complicated than an Oracle or MySQL. But it's different. It's not what people are used to. So, when you have 10 years of experience with that single machine model, there's definitely a tendency to say, “Oh, this new thing is worse, it's more complicated.” There is some of that when you move to a distributed system, but some of it's just that it's different.

[00:04:15] SK: Yeah, definitely second that. You hit the nail on the head, Jonathan. It's about inertia. In some of the customers that we have, for Cockroach, especially the really large ones that have been around for decades, they've got a very evolved and complex deployment reality.

That evolution often incorporates a quite a bit of regulation, depending on the vertical, financial services is a really good example. In a very evolved IT InfoSec posture.

So, if you think about all of those, the evolution that's occurred over decades of using databases to up and move it into the cloud, and even go step further into moving into a fully managed service where another vendor holds your operational data. It's a lot to wrap your head around. I think that's where the most significant hurdle. I think in the long run, when you kind of squint at the horizon, you see that for every company in the smallest, where I think it's most obviously useful to move to a fully managed service in the cloud, to the the largest, where there's sort of the most sort of practical impediments for everyone is going to be a better TCL to really embrace the cloud, because it really I think, lowers the costs. Your total cost of ownership is faster time to value, you can ultimately do more, and do it less expensively. So, that's just the picture that we have to paint. The reality is just that inertia is natural, and it's natural for everyone. Wrapping your head around a new database is not something you do unless you really feel like there's a big benefit.

[00:05:46] KP: Absolutely. With these cloud offerings available, you have options, like you'd mentioned, to go fully managed that rather than standing this up myself, I can hire someone who does it best in class and knows how to do it for me, and then I just am a user and pay a service fee or something along those lines. But often there is still this story of some technology groups that want to maintain the ops presence or monitoring, or maybe they think they need to fine tune the system. What do you see in practice? Are people migrating more towards fully managed services? Or are there industries that really want to keep more control on things?

[00:06:21] JE: I think that there's a mix. I think that there's growing understanding in the industry as well of how to best leverage that mix. What I mean by that is that your cloud services and cloud infrastructure, in general, you're paying a premium for those. So, you get the best return for that premium, when you have a very elastic workload, when you need to expand and contract that, and when you're on the contracting side, you're only paying for what you use.

If you have a workload that's very consistent and very high volume, then that's probably going to cost you more money in the cloud, than doing it yourself, even when you factor in all of the overheads of running your own infrastructure. So, one of the great things about distributed

infrastructure, distributed stateful infrastructure, like Cassandra like Cockroach, is that you can take advantage of that infrastructure duality, and you can deploy on premises where that makes sense, and you can deploy in the cloud where that makes sense, and you can replicate between the two in a way that you couldn't with technology from 15 years ago.

[00:07:41] SK: Yeah, those are all good points. It definitely is. We see the whole spectrum and there's huge companies that are ready to move to fully managed already. And there's, of course, small companies that are insistent that they still run it themselves. I think what we're going to see is that those distributions across the different segments of companies, they're all going to move continually towards being more eager and more accepting of having fully managed services. But they'll always remain a distribution. Just sort of as an example, company like Facebook, for the data architecture that they've built. I mean, they spent engineering millennia on it. It is not something that could ever be put into Cockroach or put into Cassandra, because they've built a custom purpose-built database, that is sort of a metadata base made of hundreds of thousands, maybe, who knows now even millions of instances of MySQL, and that is an incredible system that they built. That's not something that they would ever be able to have a vendor fully managed for them.

On the other hand, there's things that looked big five years ago, that are going to be easily within the footprint of a fully managed service. I just feel like those distributions are going to move so that there's more and more mix of the fully managed over time. The other thing I bring up is, I think, the dynamics of these kinds of industry wide changes. They have these tipping point dynamics. So, it happens – it fits and starts and you see little green shoots, and all of a sudden everyone's doing it. That's a dynamic that that we're already seeing playing out and I think it'll become very obvious, just like, for example, it's very obvious in 2021, that no matter what vertical you're in, how big your company is, why would you run your own private cloud or build your own private cloud anymore? You've got to have an incredibly specific purpose. In fact, everyone realizes, “Hey, the public cloud really has economics that are favorable.” That's going to permeate the entire ecosystem, all the way down to the database.

[00:09:48] JE: I have a slightly related take on that, which is that just as that some of the factors that you mentioned, making it more attractive to you and adopt the cloud in general, as well as fully managed cloud services. Some of those factors are bleeding in to private cloud and

reducing costs on that side as well. And I'm thinking specifically here, of Kubernetes, where even a couple years ago, you could make a case that it was too early to go all in on Kubernetes. There were some growing pains around stateful sets. There were still other companies saying that, “Hey, our technology is better than Kubernetes.” But I think in late 2021, it's fair to say Kubernetes has won and as someone who has issues with some of Kubernetes design decisions, that still makes me happy, because having a single standard is just so much more convenient for everyone in the industry.

What that means is that as we're creating Kubernetes based solutions to run Cassandra in the cloud is part of our managed service, we can also bring that technology to customers who prefer to run their own private clouds, and leverage that same Kubernetes technology to reduce their costs and improve their efficiency.

[00:11:28] SK: We see that dynamic also playing out. What we tried to do is we use Kubernetes internally so that the tools and the sort of run books that we develop, the operator, for example, is becoming very sophisticated. And that's the same one that we ultimately want to share to all of our customers that still do want to self-host. There are so many and I really do believe that's going to continue.

I think the dynamic that's going to – besides economics, that's really going to push companies that even can, have the expertise, to make their way in a private or hybrid environment with cloud is in addition to I think, what's going to be favorable economics in terms of total cost of ownership, there's increasingly this ecosystem advantage. And in the public cloud, it's not just Cockroach, for example, running a fully managed service on AWS that is easy to consume and has better economics. It's all the other things you need to build an application. It's your Cassandra. It's your it's your Elasticsearch. it's your Confluent, right? To the extent that you do that in a private cloud.

So, there's an increasing set of things. Yeah, you got Kubernetes, which makes it easier, and people will still follow that road. But there's pressure, because of the additional ecosystem advantage. When you're playing in an ecosystem like AWS, or GCP, or Azure, you get a growing list of very competent vendors, each with their own economies of scale. That ultimately, if you look at the horizon, can run that better, even than you can run it even with a very sophistic

Kubernetes operator, and all those integration points kinds of get built by the vendors in terms of their partnerships.

I think there's a gravitational pull towards managed services in the public cloud, which would be difficult even for the companies that have today and even in the short to medium term, an advantage that they feel that they can maintain by going it alone and really be making it a core competency to run a private cloud. But like I said, it's just distributions moving. You're going to see the whole spectrum. I think for companies like DataStax and Cockroach, one of our big advantages in 2021, is that we do allow hybrid, we do allow private, and that's a strategic advantage that we have over the big cloud players that say, “You're going to only use the public cloud.” And in fact, we're going to make it so that you only use our public cloud. That's not what a lot of companies are looking for.

[00:13:48] KP: Well, there is a bit of an existential threat in that regard, that maybe the big bad cloud provider could come in and offer a redundant service to what you're doing, sort of especially a threat, I guess, when there's open source components. We've seen some drama recently between Elasticsearch and Amazon and their fork and all that sort of thing. How do you guys perceive this potential existential threat to your own companies?

[00:14:11] JE: I'll let Spencer go first, because I think he's got some more interesting things to say about this one.

[00:14:19] SK: I don't know about that. I'm sure we're both noodling on this problem for years now and so is everyone in our shoes. Both of us are working on what were open core systems, and we're all trying to evolve and meet the challenges of this new reality. Amazon's done more than any other vendor to move the state of the art, but also to put competitive pressure on what I thought was a really good open core model. But hey, you either evolve or you die, right?

So, I actually think that Amazon's moves make sense in terms of how they run their business that’s why they're as successful as they are. Of course, we have to respond to that. As I mentioned, I think we actually – there are, for all of the advantages that hyperscale cloud vendor like Amazon or AWS has, there are openings, there are strategic advantages that players like Cockroach and DataStax have. One of those big ones is we're going to offer a lot more

flexibility. If you think about the market, there's an incredible contingent of companies that are looking for what are they going to use for operational data stores to build their next generation of products and services. Those companies haven't really ever been in this segment of companies that are driving Amazon success with, for example, Aurora, and DynamoDB. These are companies that are still using Oracle, and all of the sort of last generation of technology, but they're coming in mass to the cloud environments, and they're buying huge numbers of credit credits, for example, with AWS, or with GCP. These blocks of hundreds of millions of dollars that gets spent over some number of years.

Those companies are a lot more sensitive to the the flexibility that any solution that they're going to embrace in 2021, is going to offer them over the next 10 years. So, what Amazon's building on is a model that's worked very well. It's sort of the growth and sort of mid-level SMB, or commercial segments of the market. But what everyone's going to be contending for is the incredible inflow of dollars into cloud spend, and cloud platform spend, that's coming from the world's biggest companies, which just dwarf the other segments in terms of the potential. It's all that's going to grow the size of this market. So, there's a lot at stake, there's a lot at play, and there's plenty of room for companies like ours to compete against these bigger players, because we're very focused, and we have the opportunity through that focus of really providing different consumption models that appeal to what ultimately is going to be an outsize portion of this very fast-growing market.

[00:16:52] JE: Yeah, and I think besides the the consumption models which is definitely an interesting aspect of the competitive market dynamics, there's going to be some effect here on open source itself. So, the the classic, open source model, of course, I think it's fair to say that Richard Stallman and the Free Software Foundation, kind of kick that off in the ‘80s with the GPL. And then, of course, it became even more popular in the ‘90s, with Linux. And you started to see other licenses like the MPL, the MIT license, the BSD license, of course, is super old, the Apache license. But all of these had the assumption that you were either going to run this software yourself, or you were going to pay a vendor to help you with it.

So, if you're running it yourself, then you're motivated to contribute, if not patches. I think it's fairly rare, even historically, for people using Linux to contribute to the Linux kernel itself or contribute patches, but contributing bug reports like that's a form of contribution as well. You're

creating value for the project and for everyone else using it by sending a bug report or even just asking or answering questions on forums on StackOverflow, and so forth, you're actually adding value to that project. So, that's what made open source so powerful as an engine for growth and for innovation is that dynamic that either you were paying the vendor who created it, and and so you were literally paying for it that way, or you were contributing in other ways. That was your form of paying for that project.

I saw a statistic famously, a few years back that Amazon had made more money from MySQL than Oracle. That's probably true for most infrastructure projects out there. Not only are they making money from providing that infrastructure for the open source to run on, but they're standing up a fork of Elasticsearch to compete with Elastic. They're standing up. They have a Kafka service that competes with Confluent. They have a Keyspaces service that competes with DataStax, and with Cassandra. And critically, Amazon hasn't shown a whole lot of interest in contributing to these projects themselves.

It's one thing if you're coming in and saying, “Hey, we're running a Cassandra service. And by the way, here's the improvements we've made to Cassandra to make it run better on EC2 and so forth.” But that hasn't been their method of engagement so far. So, what you're seeing is a new generation of infrastructure companies are treating this as basically a bug in the licenses that they're distributing their code under. I think the first one would have been the Affero GPL. And then people looked at that and said, “Well, this probably doesn't actually fix the problem, as well as we need to.” So, then you kind of got this second generation with the BSL, the Business Source License, and the Redis Community License and the Confluent Community License, and so forth.

[00:20:31] SK: Yeah, that's actually the route that we've embraced, which is the BSL and happily, Maria DB created it. It seemed like a really good model when we looked into all of the different alternatives. Essentially, I just give a quick explanation of the Business Source License, it involves a couple unique features. You sort of start with the underlying license. In our case, it's the Apache that our core was previously licensed with. And then when you introduce the BSL, what you do is you create a term, so for how long the BSL will control, and list of exclusions on that sort of underlying license.

So, what you want, think of it as a sort of like a patent protection. You kind of protect your innovation for some period of time. In our case, we set the tournament three years, and the one exclusion we have is you're not allowed with that license to create a commercially available database as a service of Cockroach. So, it's kind of like you can't just put Cockroach into RDS if your Amazon. Of course, it's not just Amazon's anyone. And then after three years, that version, that's three years old, can then – it reverts to Apache. It always leaves open source in its wake a trail of open source. But it provides three years of protection for innovation.

I think that's one of the really crucial takeaways for me from this evolution of open source. And that's really that there's still plenty of room for it. But you just need to make some common-sense alterations, so that there's still room for innovation and the profit that can feed the innovation. The reality is, you look at Redis, you look at Elastic, you look at Confluent, all the examples that Jonathan mentioned, and all of them are doing a very wonderful, building very wonderful businesses competing directly with Amazon, for example, and others as well. I mean, Amazon also has a great business selling Redis and even more money than Oracle. I didn't know that that's a pretty interesting stat. But at the same time, none of these companies are competing with, they're doing that poorly. And I think there's that much more interesting developments to come, because of that point I made earlier, there's a lot of additional companies now entering into this new ecosystem. That's going to drive more innovation, the companies that really focus on their core product, as opposed to offering everything under the sun like Amazon, they can really create innovation that's going to be interesting at that high end of the market where a lot of the new opportunity is.

[00:23:05] JE: Right. I think it's going to be super interesting to see what happens over the next couple years to see if there's some kind of standardization around the BSL or around one of these other licensing approaches with this next generation of infrastructure.

[00:23:22] KP: Do you guys have any advice for how to start and nurture an open source project?

[00:23:29] JE: I guess it depends on what stage you're starting out at. If you're starting from zero, then step one is you need to eat, breathe, and dream about a community that you're building. It's not just about the code. I think engineers tend to have this kind of assumption that if

you write great code, then it will be its own marketing. But that's not the way it works and that's not the way it works in the either open source or proprietary software. So, the first thing you do after you have that minimum viable product is you need to be available. If somebody comes onto your forum or onto StackOverflow, or onto Discord or wherever you're directing people to ask questions, they need to get answers within a single digit number of minutes or they're going to go away. So, it's just super, super critical to bootstrap that as you religiously as you possibly can.

[00:24:37] SK: I really would second that. That's a great answer. These open source projects when you start them, they're fragile. It's exactly what Jonathan's saying. There's a short window by which you can make somebody a champion or lose them forever. I would say that there's also sort of a short window for a product. If you get too far along in a bubble, “I got to keep this thing secret. It’s going to be the greatest thing and I needed to get it really – get this impressive MVP.” I think that can be a mistake, because you want to get information from your fledgling community as soon as possible.

So, really dialed back with that MVP is, and blog about it. That’s one of the practical steps you can take to really start to find, put your tentacles out there, your feelers, find out who might be interested in what you're building, blog about it, try to get that onto Hacker News, which by the way, is priceless. If you can get on the front page of Hacker News, that will give you your initial cohort of those early adopters and innovators that are interested in what you're building. And then like everything in life, there's just this really simple way to succeed, which is focus on the small things and put love and attention into them. Never miss an opportunity. If someone reaches out to you to take the time to answer their question, no matter how foolish you might think the question is, or how off topic. Find out a little bit about what they're doing. Take the time to help them. You build these communities, one person at a time, one question at a time, you start to focus on those little things with love and attention, and the whole thing will blossom.

[00:26:11] KP: I'm curious how being open source or at least having certain open source components of your projects impacts your development cycle? Do you feel, I don't know, under more pressure, because all the code is there naked right on the repository?

[00:26:26] JE: I think it probably has less of an impact on the development cycle, than just on the mentality for lack of a better word, of doing everything in the open. This can definitely be an adjustment for people who haven't worked on open source before, to come in, and your job is to contribute to Cassandra now and you post your first pull request. And then some senior engineer who's probably at a different company, points out the things that you need to improve. So, having that very public, literally anyone on the internet can read where somebody reviewed your code and sent it back for improvements, that's a really hard adjustment for some people to make. For people fresh out of college where that's like the only experience they've had, and that's normal for them. Not so much. So again, there's definitely a difference based on what your expectations are.

[00:27:34] SK: I've definitely found, in my experience with all the open source projects I've worked on that there's value in terms of quality, and having your work available. It's kind of like there's some aphorisms. Many eyeballs make bugs transparent in open source. That's one of the early open source luminaries that says something along those lines. Also, sunlight is the best disinfectant. I do think these things are true, like to have your work available out there, it does, I think, help the average open source contributor to have some motivation, like an impetus to really care about what they're putting out there, because their name is fundamentally attached to that. Anyone can get blame that line and find out exactly what the provenance was of it, in all the comments in the code reviews and things like that.

So, I think it's kind of like this constant pressure that makes people try a little bit harder and that adds up. Every little, like everyone knows, is build a complex project of any sort, whether it's been closed source or open source, like the little bits of technical debt you create, they create a cumulative cost. You want to minimize that as far as possible. I think open source is a really strong, positive influence in that direction.

[00:28:55] JE: The other thing I really like about open source is how it kind of strips away the distractions from the technology and the decisions around that. So, I guess, there's a there's a good side and a dark side to this. The dark side is when sometimes you get in a situation where in an open source project there will be some engineer who has way more time on his hands than his good. And who really wants to go to the mat to argue something to the death. I said his,

it could be her, but it usually is a his who wants to just stall the progress on some innovation until everyone agrees to do it their way.

But the the good side of just kind of stripping away the usual interpersonal things and so forth, is that I would review the most senior engineers patch the same way as the most junior. You have that kind of democratization of the code that it's not what it says on your resume or where you went to school, it's how good is your code? How good is this patch that you send to the project? It's refreshing to have that level playing field.

[00:30:19] SK: Yeah, I like that. That really echoes my sentiment exactly. It's that idea of, it doesn't matter who the patch is coming from, put that equal effort into it. Especially give your love to that more junior engineer, and your attention, because you're going to up level them and then they're going to become a stronger contributor over time, and ultimately start to take on some of that task themselves and really scale the project. You get, in every open source project, I've been involved in, I think this is true for closed source too, unquestionably. There are certain connectors become sort of the hubs in the graph of all the different people that are contributors to the code. Those are folks who never want to lose, because they really put in this incredible effort that I can never find myself capable of matching, but of responding to virtually every pull request that comes in, they're constantly out there, trying to improve everything and to up level everyone. When you get someone like that, that's someone to cultivate, and someone to really embrace and give them as much responsibilities they're willing to take.

[00:31:28] JE: And this is one of the cool things about open source is that it really is larger than any single company. So, I can think of engineers who have contributed to Cassandra, as employees of three different companies. That's really cool that that institutional knowledge about the project can be retained across that that lifetime of work.

[00:31:55] KP: Oh, that's a very neat example. There was a, I guess, a historical perception that open source was often lagging behind, that it was the knockoff of the “real product” made by the commercial closed source team, and that they were just kind of, I know, having a lagging process of copying features or something like that. I think that's probably a straw man argument. But maybe it was true at some point. What do you see is the current state, why is open source

really been a leading, a flagship path that a lot of systems are taking, rather than being closed source?

[00:32:28] SK: I can offer my perspective just on databases, when we started building Cockroach DB, I guess, seven and three quarters years ago. My reaction post Google in 2012, when I left and looked at all the different things that were available for operational databases, was just that, boy, of all the things out there, I'm certainly can use open source. Because databases can be pretty complex and if you want to build something really ambitious, not being able to rely on that larger community, not being able to go in there and sort of debug your own problems, it's almost a non-starter if you really have ambitions, and that was really the position of people at Google. Just because no matter what open source project Google might bring in and use and really scale, they were going to be issues. I mean, Google was busy trying to fix things in the Linux kernel, right? That was pretty interesting example.

But for them not having open source, anything that they were forced to use closed source, it's just a much more difficult process to really get it to exactly what they wanted. That's always been my approach. So, open source, for me, was the only viable path to actually building a database that could succeed in the late 2010s and now going into the 2020s. I think that if you queried developers, for example, that are kind of in that early adopter, innovator, part of that crossing the chasm bell curve, those are the folks that are often on Hacker News. I bet if you surveyed them, and you ask them, whether their comfort level is higher with open source than closed source, you get a pretty definitive answer. That's ultimately why it's open source that's leading the charge now. It's the faith that developers sort of worldwide put into open source software. It's just a better model. And I think people recognize that.

[00:34:27] JE: Yeah. I think there's a couple just market dynamics that are contributing to this as well. If I'm starting a company around – and this is specific to the infrastructure space, right? If I'm starting a company where I'm saying, "I'm going to build a better message bus, and you should trust your data to my new product, I'm going to need an A round of hundreds of millions of dollars to get that product to the point of maturity it takes to get awareness in the marketplace and to get people to have that level of trust in that proprietary product." Contrariwise, if I start that as open source, the way and since I'm talking about message buses, the way Apache Kafka was by the founders of Confluent, or Apache Pulsar was by the people at Yahoo, then I

can use that as a growth hack. It's an adoption and a marketing mechanism, where companies are much more comfortable deploying an open source project that said, it's a one dot O project, or maybe it's an even an O dot nine project. Because they recognize that.

Worst case scenario, if I have to, my engineers can figure this out, and how to fix bugs in this, or how to migrate me off of it, if it comes to that. So, there's that security and that assurance that lets companies adopt your project or your product faster than they would with a proprietary approach. And then the other thing is that you're just like, one of the things that I was very serious about when I started DataStax is building a remote first engineering culture. Because there are some really, really smart engineers in Google and in the Bay Area in general, right? But if we compare all the smart engineers in all of the Bay Area versus the smart engineers in the rest of the world combined, like the rest of the world wins, hands down. So, there's a bit of that dynamic in open source as well, where it's like, do you want to bet on the engineers building the Solaris Kernel? They're super smart guys not taking that away from them, but do you want to bet on them, or do you want to bet on everyone else in the world contributing to Linux?

[00:37:00] SK: Those are great points. You knock something loose in my memory and I wish I remember the gentleman I was talking to. It was someone that was advising me early on when building Cockroach Labs. He made this point that there's information asymmetry available to companies that are building open source products or open core products, and that by having that community that starts so early and that early feedback, and outside contributors, and one thing that you always see when you have an early open source project is some huge company or some champion at a huge company, it's like, “We want to use this now.” And it's like, “Well, it's an alpha. It doesn't quite work for you yet.” But that person can immediately give you all kinds of insight into what's going to make your product wonderful if it could do this, this and the other thing. You don't get that when you're building in a bubble in a closed source environment, or it's much more difficult. You're out there trying to talk to people and educate them, you don't get to blast out on Hacker News to 10,000 people, of which some small percent, but it actually adds up to an absolute number that's meaningful in terms of information that's really targeted about how you can build the best product to actually meet the market.

So, there's an information asymmetry advantage available in the open source model that I think allows you to build better software faster.

[00:38:17] JE: That's a really good point. Engineers working on open source have – there are like two clicks away from getting direct feedback from people using that software, versus my experience, at least in the proprietary software world is that's usually filtered through several layers of support and product management.

[00:38:41] KP: So, I'm curious, when commercialization of open source was first introduced to me, it was the company Red Hat. They built a very popular Linux distribution. I guess, if you're a developer evangelist at the company, you'd say it's the best Linux distribution, and they gave it away, and then they were going to sell services on top of that. You come to us when you need the people that built it to help you run it or things like that. The unit economics of that, I guess, worked out pretty well for them. Could you compare and contrast how that model compares to your own companies?

[00:39:14] JE: I mean, I think at a super, super high level, DataStax wants to be the Red Hat of Cassandra, right? I think that's definitely the example that most people point to, when they're talking to venture capitalists and so forth of Red Hat is like the poster child for creating a successful business on top of open source.

I think that the biggest difference, I would point to that's different in 2021, versus, gosh, I don't remember when Red Hat was founded, but it was in the ‘90s, sometime. The biggest difference now in the infrastructure space, is the prevalence of the cloud and so that includes both the Kubernetes layer that allows you to deploy across private clouds in a standard way, but it also includes the managed services that you can build as one of the innovators around an open source project. I think that is the main way more than the classic services and support model, the main way to fund open source in the future is these managed services. That's why coming back to the earlier point, that's why it's so critical to get the licensing model right in a way that allows the companies doing the innovation to continue to fund that innovation.

[00:40:47] SK: Yeah, I'll try to add. Maybe I'm just going to echo your point, exactly, Jonathan, because I think you made all the right – you touched on all the right things. Basically, I think the

reason open source had such an ascendancy in the aughts in 2010s, maybe even the late ‘90s, is because offered a faster time to value. I mean, adoption of any software ultimately is going to come down to its utility. And developers, if you think about the closed source model previously, they would get something at a conference or in some trade mag and they'd be interested in it. And then some, get some rep on the phone, and they'd come visit you and they'd explain and educate you. And then you'd have to go through procurement, you'd finally get the thing, and maybe needing some machines to run into this every requisition. This thing could take months. I mean, literally, could easily take six months, depends how complex the software was.

Of course, then you have to learn how to run it and everything and use it. With open source, all of a sudden, you had – by the way, you got printed manual sent to you in the mail and CDs and things. So, open source, of course, with the rise of the sort of early cloud and the HTTP, and so forth, being able to surf the web, find these communities, the things that have morphed into StackOverflow over the years, but you used to have Usenet groups and things, all of that provided a much faster time to value. All of a sudden, now as a developer at an organization trying to make some product or service a reality, you were able to say, “Hey, I need this, this and this other thing.” They're all open source and go read about them, I have questions I put them in, I get a response in an hour. The responses are already there when I search on Google.

So, that was just such an obvious win. You could shave months of your time to get something valuable that you could either show to other folks in order to get funding, or actually just build and deploy, which is another very common pattern. What we're seeing now is a move to even faster time to value. It's why these open sources is again evolving. It's certainly not going away for the reasons of community transparency, speed of evolution, information asymmetry, marketing. I mean, open source is a wonderful marketing channel. People people have a good feeling about it. They like the idea that they can get in there and understand the ideas if they need to.

All of that is going to keep open source fair. But it's really how does it further evolve in order to create faster time to value. And here, you realize that a fully managed service, especially one that has totally free tier that you can start on, that can get you really a long way before you might have to put a credit card down, this actually means that not only can you access that community and embrace all the benefits of open source, but now you can eliminate the need to

learn how to run the software. I've used that to learn how to use it, for example, with the database. You got to learn how to – what are the differences in terms of the ORM that I have to use, or whatever it is. How do I integrate this with my application? But now you don't learn how to set up alerting and how to properly deploy and configure things in complex ways.

So, now you say that, hey, instead of getting started in, say, several weeks of trying to learn how to run the different pieces of open source software that you have to deploy into containers and on VMs out there in the public cloud, but it's like, “Okay, I'm actually just going to write my application and this whole thing's going to run.” Because other folks already know how to run this and actually have a very competent way of running them. The time to value has decreased. So, open source coupled with fully managed services that have free tiers, it's kind of like all the best aspects of open source, but even faster time to value. It seems pretty clear to me that that will be the future.

[00:44:21] JE: Exactly.

[00:44:24] KP: As a software engineer, I really don't want to ever have to worry about the compiler. Same as I don't want to have to worry about the electric or the water coming into my house. Maybe if I'm a head’s down application developer, and I'd like to take some of the same approaches to adopting a distributed database, I'd love to just benefit from the right product choice. But under the hood, I know there's things like the CAP theorem and maybe the Paxos protocol going on. To what degree should a developer educate themselves about distributed systems before working with a distributed systems database?

[00:44:54] JE: Man, that's a really excellent topic there. As a database nerd, it hurts me to say this, but a lot of people want the database to be like the compiler, like you said. They just want it to work, don't want to have to think too hard about dash O2 versus dash O3, or whatever. Similarly, on the database side, a lot of developers just want to put their data in and get their data out and move on with their day.

So, one of the things that that I think, is going to come out of this, and then I'll come back to what I think the flip side is, but one of the things that's going to come out of this is that, people just want to get their data out with a REST API, they don't want to install an ORM, they don't

want to install a traditional FAT client driver. They just want to make an HTTP call, whether it's rest, whether it's graph QL, and that's how they're going to interact with their database.

The flipside is, I said much earlier that the more you care about performance, the more you do need to understand how the system works under the hood, and that's not going to change. I don't see how it could change. So, there is always going to be that need for people who have that next level down understanding. What happens when I do a join that has to hit multiple machines? And why is that significantly smaller than a query that only hits a single partition, and so forth.

[00:46:34] SK: Yeah, I think performance is the critical sort of threshold. Once you start to care about performance, then you need to learn more. I think before that, there's an opportunity to really treat a lot of the things out on the market, whether it's Mongo or Dynamo, or Cassandra, or Cockroach or Aurora. You can squint, and they're all going to pretty much do what you want, especially if they offer simplified interfaces to Jonathan's point. I think that there's kind of an approach that can be very helpful for developers that are trying to build something and don't yet know all the ins and outs that would allow them to make a truly informed choice about what's the best software for the task at hand. That's association.

So, what are other people doing? What are some of those use cases that you can pattern match with the thing that you have in mind that you want to build? Where are your customers? Are they geographically distributed? How much scale are you planning to have? What sort of features do you have? And then you look at a company that has something similar that they built, or another open source project, for example, that you can kind of look in and say, “What are the choices?” There are lots of blog posts that companies put out, and you can just pattern match there and find something that's relatively close and say, “Well, what did they choose? Do I think they're a good set of engineers that are making good decisions?” And sort of part and parcel with that when you actually look at what company like Cockroach or DataStax needs to do. It's really to build reference architectures.

You have, let's say, a forgivable repo that uses Cassandra, that gets something done that's useful, and they could just fork that code and actually start to build on that. And they know that, hey, this is already something that is a reference architecture, that's well tuned to what

Cassandra is going to offer. And so, I can run with that and feel pretty confident that both the company understands my use case that they can support me, that this use case will definitely work with Cassandra, for doing something that's very close to what I need to do.

So, I think that's a good approach for people that – because it's a, to Jonathan's point, about being a database nerd, I guess I'd become one too. You can spend years understanding these things and still realize you got a lot to learn.

[00:48:48] JE: Yeah, so I think the difference between good product design and bad is that with a good design, you can let people be productive with a set of knowledge that's appropriate to what they're trying to accomplish. In other words, if I'm trying to build a demo on my laptop, I should have to understand – the requirement of what I need to know about the system should be much lower than if I'm home depot serving 100,000 requests per second out of my production cluster. So, having that learning curve where you can get to the next level when you need it, and not have to preload that to get your first hello world done. That's a good design.

[00:49:33] KP: Well, speaking of hello world, maybe to wrap up, we could give the pitch for what is the use case that typical developers and companies find themselves having that leads them to explore Cockroach DB, and also DataStax as the tool of choice?

[00:49:50] SK: So, for Cockroach, I'd say, the really killer differentiator that we've put a lot into, well, there are two. There are high scale and we have a theory just that there's sort of analytical big data. But now there's transactional big data, because it's not just humans on desktops or humans on mobile devices. It's now virtual and IoT type things that are all hitting APIs, which ultimately did an operational database. So, there's that data intensity, and then there's also, you know, nowadays with global app stores and global public cloud, anyone could build a product or service that could reach people in Brazil as easily as it can reach people in Australia. And you want that to work equally well. So, really building for that geographic reach is something that we've invested heavily in with Cockroach.

Of course, both of those are within the context of do you need a SQL database. So, I think that's sort of the killer feature. You need those differentiators, and your preference, whether it's institutional muscle memory, or it's, we really want to have this sort of SQL as our query engine,

or we need to have ways to explicitly manage the data model, all those things would bias you towards using SQL. So, if SQL is your product category, and you have a need for those differentiators, that's the sweet spot for Cockroach.

[00:51:12] JE: Yeah, Cassandra was was created to solve problems of performance and scale that SQL databases couldn't tackle. Certainly, Cockroach is solving those in a different way. So, the distinction I would make there is that Cassandra places more emphasis on performance to the point where our default isolation level to use a relational term, the default isolation level is just read everything. So, you can opt in to more strict serialization, but Cassandra's emphasis is on. I want to do new high performance, thousands or hundreds of thousands of operations per second across anywhere in the world.

[00:52:06] KP: Very cool. Well, Spencer and Jonathan, thank you both so much for taking the time to come on Software Engineering Daily.

[00:52:12] JE: Thanks, Kyle.

[00:52:14] SK: Kyle, it’s my pleasure.

[END]