EPISODE 901

[INTRODUCTION]

[00:00:00] JM: Cassandra was initially released in 2008 as a project out of Facebook. Cassandra offered an open source solution to database scalability issues that were being tackled internally by large companies such as Amazon, Google and Facebook. 2008 was a golden age of new infrastructure with systems such as Hadoop and Kafka gaining traction around the same time.

Jonathan Ellis started working with Cassandra and became intrigued by the system. With the help of investors, Jonathan began working on a company based around Cassandra called Datastax. Today, Datastax is a company with more than 450 employees and a large valuation. Jonathan joins the show to discuss his experience working with Cassandra and his reflections on the becoming of an entrepreneurial founder of a highly successful database company.

There's a wealth of useful knowledge for both software engineers and entrepreneurs in this episode, and it was a real pleasure talking to Jonathan. He's a true engineer's engineer as well as entrepreneur's entrepreneur. So, quite an awesome episode.

[SPONSOR MESSAGE]

[00:01:16] JM: As a company grows, the software infrastructure becomes a large, complex distributed system. Without standardized applications or security policies, it can become difficult to oversee all the vulnerabilities that might exist across all of your physical machines, virtual machines, containers and cloud services.

ExtraHop is a cloud native security company that detects threats across your hybrid infrastructure. ExtraHop has vulnerability detection running up and down your networking stack from L2 to L7, and it helps you spot, investigate and respond to anomalous behavior using more than 100 machine learning models.

At extrahop.com/cloud, you can learn about how ExtraHop delivers cloud native network detection and response. ExtraHop will help you find misconfigurations and blind spots in your infrastructure and stay in compliance. Understand your identity and access management payloads to look for credential harvesting and brute force attacks and automate the security settings of your cloud provider integrations. Visit extrahop.com/cloud to find out how ExtraHop can help you secure your enterprise.

Thank you to ExtraHop for being a sponsor of Software Engineering Daily, and if you want to check out ExtraHop and support the show, go to extrahop.com/cloud.

[INTERVIEW CONTINUED]

[00:02:48] JM: Jonathan Ellis, welcome to Software Engineering Daily

.

[00:02:51] JE: Thanks, Jeff. It's good to be here.

[00:02:53] JM: Cassandra was initially released in 2008. What were the challenges at the database layer that large internet companies were dealing with back then?

[00:03:02] JE: I'm going to get into my way back machine here for second. So, I started working on Cassandra as kind of as the second generation of developers, because the first generation was created at Facebook and I wasn't part of that. So I came onboard when it was open source in the summer of 2008, like you said. And the way I got to that was I had been working at a backup provider called Mozy for 3 or so years before that. And this was kind of early days of web 2.0 is I think what they were calling it at the time.

Kind of the realization happening in the industry that we were going to be building these connected applications that were web first, and later on you had kind of a corollary of mobile first. But the common thing they hadn't had was that instead of deploying your software to an organization, the classic, "I'm going to have my exchange server running onsite or I'm going to have my Oracle ERP software running onsite. I'm going to have this cloud application that the entire country is going to be my user base."

So we encountered that at Mozy that we actually built a custom storage system for the backup data itself, for the pictures and videos and so on. But then we had a need for a database to keep track of which files belonged to which users. And that was actually a big problem at the time, like the state-of-the-art was a sharded Oracle or a sharded MySQL, which comes with a lot of problems in terms of both reliability and availability, but also performance and the ability to scale.

So I've recognized from personal experience that we were entering this new era of application development and the database was a bottleneck and kind of an unsolved problem where the state-of-the-art was not a good place. The sharded relational was not a good place and we needed something better.

So I actually talked to the Rackspace labs team at PyCon. I gave a talk on the work that I've done for Mozy building their storage engine, which I've actually done in Python. And we can talk about that. Actually, it's a fascinating topic of why I did that and why wouldn't do it again that way. But we talked a little bit and I said, "Hey, I think that the next really interesting problem to solve is scalable databases." And they said, "Well, actually we need that as well. We're starting to run into that problem at Rackspace. We'd love to hire you and have you help us solve that problem."

So, I got to Rackspace just a couple months after Facebook released Cassandra's open source, and I got to evaluate. At the time there was MongoDB. There was HBase. There were some systems that kind of faded away like Voldemort, and Dynamite was another one. And so I got to evaluate kind of these nascent scalable database projects, and I really thought that Cassandra had the best combination of – It has the robust data model as supposed to some of the others that were only key value.

But Cassandra also had a fully distributed model that made it much easier to build in the kind of availability characteristics that you need when you're building cloud applications. So that's the really long answer of how I got started.

But, yeah. So, I think it was clear to engineers at the time that this was a problem that needed solving. But coming back to your original question, I don't think it was nearly as clear to nontechnical people. So, actually when we were fundraising for Datastax, the most common response we got was, “Yeah, this is interesting technology. But there's five companies in the world who actually need a scalable database.” Of course, today, there is thousands of companies running Cassandra. I think it's clear that it's a ubiquitous problem.

[00:07:23] JM: Well, that is what happens. You start out with the five companies that need it and then over time you just have the growth. If we talk about the literature at the time, Google had published the Bigtable paper. Amazon had published the Dynamo paper. These things contributed to the architecture of the Cassandra model. Although Cassandra, I believe also had some novel implementation details. Give me your recollection of the state-of-the-art of distributed data systems back then.

[00:08:01] JE: Yeah. That's right. That's a good summary that Bigtable and Dynamo are definitely the most well-known. If I remember correctly, Bigtable, the paper was published in 2006 and Dynamo in 2007. I think in a lot of ways it's not wrong to think of Cassandra as Dynamo 2.0. So, at least one of the engineers who invented Cassandra, and possibly both of them, I don't remember for sure. I was one of the authors of the Dynamo paper from Amazon.

So, Dynamo – Well, I'll backup and I'll do it in chronological order. Bigtable says, “Okay, we have at Google a distributed file system called GFS, and it’s the spiritual father of HDFS in Hadoop world.” So Google said, “Given that we have a distributed file system, how can we build a database on top of this?” So that's where Bigtable came from, is it's designed to run on top of a distributed file system, and that kind of takes care of the storage aspect of it. The file system is in charge of the replication. Then the databases is just – Just. I put just in quotes. It's still a hard problem. But the database is just in charge of mapping chunks of storage to tables of data that the users have.

So, it's actually fairly complicated system where you have master of the Bigtable system and the you have tablet servers that are each responsible for ranges of data in the tables that you're serving, and both of those are single points of failure and the need to have failover. To handle that failover, you have another service that was Chubby and the Bigtable paper. That's their lock manager. Then you have the surface area of GFS itself. It’s very impressive. I'm definitely not trying to take that away from you, but it’s also – There’s a lot of moving pieces.

So the Hadoop ecosystem or the Hadoop community said, “Hey, we’ve got HDFS that’s very similar to GFS. We can build Hbase that’s very similar to Bigtable.” So, as I said, that was one of the options that I looked at when I was evaluating the landscape. But at the time, Hbase was about 250, 000 lines of code if you included some of those dependencies. And Cassandra was about 35,000 lines of code. So part of my thinking was as a fairly small team at Rackspace, how comfortable am I going to be fixing a problem that some interaction between HDFS and one of the tablet servers and possibly ZooKeeper. That a lot of surface area.

[00:11:00] JM: You’re afraid of Hbase.

[00:11:01] JE: Yes. So, the complexity was one of the things that scared me away from that. But then in 2007, Amazon published their Dynamo paper, which is a very different take on the same problem space, and they said, “As supposed to Bigtable, Bigtable gives you keys and it's almost like a map of maps where you have kind of partition keys, but for each partition you can have a key value map associated with that.” And they call those columns, which is confusing, because in the relational world, a column means like a very different thing. So, there's definitely some terminology confusion that we did at Cassandra. We followed the Hbase –

[00:11:43] JM: Yeah. You got it right. You got it right eventually.

[00:11:46] JE: We got it right eventually. But Dynamo says, “Hey, we’re not going to do maps of maps. We’re just going to do key value. And what we’re going to do is to allow you to mitigate between competing updates to the same value. We’re going to use this too called Vector Clocks, which basically gives each update to a value a unique identifier that's associated with the client that made that update and they’re order logically, not chronologically. So even if the clients doing the updates are – Their clocks are completely out of sync on the physical machine. Since it's a logical clock and not a chronological clock, we can still detect what order those changes were made.

Then the problem with that system is that it then kicks out to the application developer, “Hey, there was this conflict, these two concurrent updates.” Now it's up to you to figure out what to do with that and to decide what the combined value should be.

So when those engineers went to Facebook and started building a next-generation system, one of the lessons they took away from Dynamo was, “Vector Clocks are too hard for application developers to use. So we’re going to do something simpler. We’re going to split those key values apart. We’re going to use the Bigtable-like data model of a map of maps.” That reduces your conflict surface area significantly, because now each client can use a different sub-key within that partition that you're distributing. And so you can avoid conflicts by using unique sub-keys like UUIDs as your sub-keys across those clients. Then if you do have conflicts even within that split out partition, then we’re just to use simple timestamp-based reconciliation, which is technically less elegant, but also much easier for average developers to reason about.

[00:13:46] JM: So, if I understand correctly, those selections that the Cassandra engineers made. That led to the tunable consistency model, right? Because you kind of have these options, these fallback options, right? Was that novel in the Cassandra world? I don't know to what degree the consistency model of Dynamo was also tunable.

[00:14:09] JE: Dynamo was also tunable. So that was the other big difference from Bigtable, was Bigtable says, “I have leaders for these ranges of data, and each leader is the master of that range. If I lose them later, then I need to do a failover and elect a new one and the data is unavailable until that new one is elected.” Whereas Dynamo saying, “We’re not going to have any leaders. It's going to be completely distributed. Any replica can update its own data without having to consult anyone, which makes it much more robust of failure, but now it also introduces the possibility of inconsistency were different replicas have different answers.”

So that's where the tunability comes from, is as a client, when I'm making an update, how synchronous do I want to require that to be across the different replicas. That was that was kind of where it started, was as you can choose a quorum of replicas or you can insist on all the replicas, which obviously makes it much more fragile if something goes down, or you can just say, “Hey, just one replica is fine, and I will tolerate the milliseconds of inconsistency.”

Then later on in Cassandra's lifetime about like four or so years later, I contributed kind of a second option based on Paxos that we call lightweight transactions. So that actually lets you do serializable operations in Cassandra, which you wouldn't be able to do with just the quorum or eventual consistency options.

[SPONSOR MESSAGE]

[00:15:56] JM: The O'Reilly Software Architecture Conference is coming to Berlin November 4th through 7th, 2019. I've been going to O'Reilly Software Conferences for the last four years ever since I started Software Engineering Daily.

O'Reilly Conferences are always a great way to learn about new technologies. You get to network with people. You get to eat some food, and there's no better type of conference than a software architecture conference, because it's a great high-level explorational topic. And on November 4th through 7th, the O'Reilly Software Architecture Conference is coming to the City of Berlin, and you can get a 20% discount on your ticket by going to oreillysacon.com/sedaily and entering discount code SE20.

O'Reilly Conferences are a great place to learn about microservices, domain-driven design, software frameworks and management. There are lots of great networking opportunities to get better at your current job or to find a new job altogether. I've met great people at every O'Reilly conference that I’ve gone to, because people who love software go to O'Reilly Conferences.

You can go to oreillysacon.com/sedaily to find out more about the software architecture conference. These conferences are very educational and your company will probably pay for it, because you're going to learn about how to improve the architecture of your company. But if they don't want to pay, then you can pay, and you can get 20% off by going to oreillysacon.com/sedaily and use promo code SE20. That link is also in the show notes for this episode.

Thank you to O'Reilly for supporting us since the beginning of Software Engineering Daily with passes to your conferences. Thanks for producing so much great material about software engineering. I particularly enjoyed the episode that I did with Tim O'Reilly a year or two ago. And in that episode I really got a better understanding of how Tim O'Reilly built his conference business. So you can always check that episode out.

Thanks again to O'Reilly.

[INTERVIEW CONTINUED]

[00:18:14] JM: I want to take a moment here to reflect on the fact that you fell into this world of distributed systems theory, and that field is not for everyone. So, like I've mentioned a couple times on the podcast this distributed systems class that took in my last year of college. I could've failed that class. I could've failed that class and then not graduated, and it was the most stressful class I've ever taken. It scared me.

Frankly, it scared me. It scared me as a computer scientist, because I was like, "I don't think I can deal with this stuff. It's too complicated." You mentioned vector clocks. That was one of the things that we dealt with and I was like – I mean, you can get grasp of them, but it was just hard, and programming the stuff was super hard. The programming projects were super hard.

I mean, I have a lot of respect for people who actually thrust themselves into this industry, like the distributed systems both applied and theoretical. I mean, that class was more emphasis on the theoretical, but then we had to apply them in these projects, and it was just brutal.

Some people like actually get a fascination out of this. They actually enjoy it somehow. Did that happen to you in college or did that happen to you after college when you were working in the industry?

[00:19:31] JE: Yeah, I think you hit the nail on the head when you said I kind of fell into it, where I'm going to digress just a little bit and talk about the virtues of startups. Because the thing about a startup is – And this was true 15 years ago. It's may be less true now because there's a lot of really, really well-funded startups now. But certainly there is a category of startups where they've raised a few million dollars and they have a very clear understanding of, "We need to get to our next milestone of success, because the money will run out in 12 months and we need to be able to demonstrate that level of success to raise the next round or to do whatever makes sense after that."

So, startups are kind of looking for – In an ideal world, Josh Coates at Mozy would have hired a distributed systems engineer out of Google or out of Amazon and said, “Okay, I want you to build our storage engine, and I am confident that you can do it because you’ve done it before.” But those engineers would have cost probably 3 to 5 times the salary that I was looking for.

So he’s like, “We’re going to take a risk. We’re going to hire Jonathan Ellis to do this.” And he kind of has no business whatsoever, because he's never done this before. So we’re taking a big risk by doing this, “But we’re going to roll the dice, and we think he's smart. We think he can figure it out.”

So startups to a large degree, I think, are willing to make that tradeoff if you can make a good case for why you think you can figure out their challenges with them. They're going to be open to letting people try new things that on paper they're not qualified for in a way that larger companies I think are more reluctant to do. So that ended up working out really well for me and it turned out that I really enjoyed it.

I would agree with what you said in the sense that the distributed systems theory is ridiculously difficult. And I think of Leslie Lamport's paper called The Part-Time Parliament, where he introduces Paxos. That paper makes my head spin.

[00:21:40] JM: That was originally rejected from ACM, I believe.

[00:21:43] JE: You could be right. I'm not sure about that detail.

[00:21:47] JM: Well, that's the one where he talks about the island, right? Like the whole thing is an allegory.

[00:21:52] JE: Yes, and he's got the – Yes. But then in the –

[00:21:57] JM: The cheese inspector.

[00:21:58] JE: Yeah. 20 years later or something, and maybe it was – I think Part-Time Parliament was published in the 70s. But then in the 90s, like you got impatient with like all these mere mortals who couldn't understand how brilliant this idea was. And I'm not being sardonic there, like it really was brilliant.

[00:22:16] JM: That's right.

[00:22:17] JE: He wrote another paper called Paxos Made Simple and he’s like, “Look, I’ll use smaller words this time around.” That's the paper that I recommend if you’re interested in Paxos or in distributed systems. It's one of my favorite papers and it's significantly easier to understand.

So, I think of myself not as a distributed systems theorist, but as a distributed systems engineer. If you explain something to me, I can go build it. I think that was something I was pretty good at.

[00:22:47] JM: Yeah. I mean, I did so bad in that class, and someday I want to reach out to the professor, because he was such a brutal professor, but it really helped. He even said in the class, he was like, “This class is going to be brutal, but you’re going to thank me later on in your career,” and it’s just like absolutely true. I'm very thankful for the brutality. But I digress.

Actually, did Cassandra ever move to Raft? Did you guys update to Raft or you stuck with ZooKeeper, and that's Paxos?

[00:23:16] JE: It took us long enough to debug Paxos that we didn't want to debug Raft on top of that.

[00:23:20] JM: Oh! So, you didn't you didn't use ZooKeeper.

[00:23:22] JE: No. No, we built our own Paxos implementation.

[00:23:25] JM: What was the reasoning there?

[00:23:27] JE: Again, we didn't want to introduce an extra dependency, and also ZooKeeper is very opinionated around we’re going – It does what's called multi-Paxos, where it elects a leader and then that leader’s responsible for doing a bunch of transactions until it dies. But when it dies, then you have that period of unavailability again.

So, Cassandra actually uses more basic version of Paxos without that leader election. So that as long as you have a quorum of replicas available, you will be able to process your transaction without waiting for an election.

[00:24:06] JM: Okay. I don’t know if you saw this, but I think Kafka recently factored out ZooKeeper.

[00:24:14] JE: Yeah, they published a proposal or a statement of directions saying, “Hey, we’re going to do this.” So I don't believe there's a proof of concept yet, but they’ve basically came to the conclusion that ZooKeeper doesn’t – Again, kind of the same place we were at, which is ZooKeeper has these opinions about the way consensus should work. And this isn't actually the best fit for what Kafka needs. So we’re going to build something that's a better fit.

[00:24:38] JM: It sounds like a moon landing level engineering effort.

[00:24:42] JE: I would say yes, but the thing is that the moon landings of the early 2000's in computer science are the routine development efforts of 2019. Unlike I think the actual rocket science, I think in computer science we’ve done a really good job of kind of extending the state-of-the-art in saying, “Oh! This is actually a well-understood space. It was really difficult in 2005, and now we have a really good understanding of how to solve that problem. And maybe even more important, we have a really good understanding of how to test solutions to that problem and convince ourselves that a new consensus implementation is actually solving the problem that it purports to solve and we’re confident about that.”

[00:25:34] JM: Is it like a broad trend of your furniture, or you’re referring to like the ability to use Kubernetes? Like we could just use Kubernetes and now we have really good programmable abstractions for doing distributed systems.

[00:25:44] JE: Yeah, I think it is a trend across the industry that these things the required Ph.D. level, research and development efforts for Amazon in 2007, that's something that you just import Raft today or you spin up Kubernetes instead of building complex homegrown deployment system.

[00:26:09] JM: Yeah, or like stringing together some scripts or something. How long did it take for people to start trusting Cassandra enough to actually use it as critical infrastructure? When did it become actually like an abstraction that people were demanding? I think Comcast was the first. That was like the first big customer that wanted help, right?

[00:26:31] JE: This is like 2011-ish. So, my memory is a little bit fuzzy here. So Cassandra was open sourced in 2008. It was contributed to the Apache Foundation either at the end of the year or the beginning of 2009. I became a committer on that Apache project in like March or so of 2009, and then I started Datastax in May of 2010.

So, we were already getting – By the time I started Datastax, we already had people using Cassandra in production. This was like Cassandra version 0.5 or 0.6. So it was definitely at the – We're moving very fast and things are changing very quickly. Again, a lot of them were startups, were looking at this and saying, "If have to build a data infrastructure layer the old-fashioned way with sharded MySQL, that will break my company, because I just don't have the six-person ops teams to keep that running." So, they're willing to take risks on this new technology, because the devil they know is just not sustainable for them.

So I remember, Digg was one of the very earliest to use Cassandra in production. If you remember that early social media site.

[00:27:52] JM: Of course.

[00:27:53] JE: I don't think I can tell you who was actually the very first, but Digg was one of the first. There was a company in Austin called OneSpot that was also very early on. Those are the two names that jump out to me from that time period.

[00:28:06] JM: That sharded My SQL. If I recall, the difficulty there was that the client had to handle the sharding, right? If you want to write your – Not Ruby on Rails. I guess it would've been – What? Like basically your Java application, right?

[00:28:24] JE: Yeah, it was all about the LAMP stack back then, right?

[00:28:27] JM: Oh, LAMP stack. Okay. So your PHP application would have to know what shard to address to get the right piece of data, right?

[00:28:34] JE: Exactly. Yeah. So you'd have to build a custom router service in front of it perhaps, or build it into the client. Those are kind of your options.

[00:28:43] JM: Which is just brutal, and an application developers should not have to deal with that.

[00:28:47] JE: Right. So that's the problem on the application, but then on the ops side, like MySQL replication would just fall over and then it's like, "Okay," and you need a human to intervene and clean this up. In the meantime, I'm down to – If I have just a single master slave pair. If I lose the other machine while I am fixing that replication, then I've lost that in production, and that's really scary.

[00:29:10] JM: Was Oracle solving that problem? Could you just buy Oracle and like be okay?

[00:29:16] JE: I never worked at an Oracle shop at the level needed to say, for sure. Certainly, the price alone was a big deterrent if you were looking at solving it with sharded Oracle.

[00:29:28] JM: Yeah. It's just interesting that problem didn't get – Well, I don't know. I guess because it ultimately was like startups. Companies with no money were like the ones who were just like, "We need some kind –" Well, I don't know. We'll have to do a retrospective –

[00:29:38] JE: Yeah. This was kind of the golden age of open source too where MySQL and PostgreSQL had kind of demonstrated that open source infrastructure can actually be just as reliable as paying thousands of dollars for an Oracle license.

So, I think by the time that Cassandra came along, that ground had been broken and that wasn't – I wouldn’t say it was not an obstacle, but wasn't nearly the obstacle that it would've been 10 years previously.

[00:30:05] JM: So then when you look at today and you have companies that can actually solve sharded MySQL, like we’ve Plan Scale on the show a couple times. Is Cassandra solving a different problem now? Because like now I feel like the Cassandra ecosystem solves maybe a different set of problems.

[00:30:24] JE: I don't want to speak for the entire Cassandra community. But at Datastax, I think our emphasis has shifted a little bit to what I would call the hybrid cloud problem. I don't want to do too much of the Datastax sales pitch, but this has some relevant statistics. We have roughly 50% of our customers using Amazon's cloud, 25% using Microsoft's, 12% using Google's, 12% using another cloud. Something like 60% to 75% also running our software on their own infrastructure. So if you add those numbers up, that's well over 100%.

So, we’re motivated to know solve the problem of not just distributed in the end, “I’m going to partition my dataset sense,” but the distributed in the my data lives in different places sense. I think that's a more difficult problem and a more interesting one.

[00:31:25] JM: Interesting. So just the heterogeneity of where your infrastructure is sitting.

[00:31:31] JE: Mm-hmm.

[00:31:32] JM: So what have you learned in the years building Datastax, and particularly when the focus has been on Cassandra. What kinds of like lessons have you learned that have given you a core competency to feel like there's like a business opportunity. There's an engineering differentiation opportunity. Like, how are you thinking about the strategic as you begin to change that strategic focus? What's motivating that?

[00:31:58] JE: I think it's not a secret in the infrastructure space that the public clouds are – They’re a friend, but they’re also foe. What I mean by that is – I said a couple minutes ago that 2005, 2008, this was that the golden age of open source. I don't think we’re in that golden age anymore, because when a vendor like Amazon comes in and says, “Oh, hey! Thanks for creating Apache Kafka, people at Confluent. That was a very public spirited of you. Now, I’m going to create Apache –” I don't remember what the abbreviation is. Is it MKS, Managed Kafka Service? Something like that? They did the same with Elasticsearch.

There was this golden age where companies were saying, “Hey, we understand there is this very brief window where it says, “Oh! There's this clear model for how to build open source infrastructure and create a business on top of that while also giving away this IP in the open source project.” I don't think we’re living in that world anymore. So you have Confluent and Elastic and Redis and MongoDB.

They’ve all changed their licenses to – I believe, all of the examples I listed would not be technically open source under the definition of that term even though the source is available and it's free to use for people who want to deploy that on their own infrastructure. But the license is set up so that Amazon can’t just say, “Thank you very much for being our R&D department, and now we’re going to go sell this and compete with you.”

Datastax has taken a slightly different approach, and I think we got started on this a little bit earlier. We started doing this in 2016, where we said, “Hey, we've had this model where we’re going to contribute all of the core database improvements to Apache Cassandra, and then we’re going to build our business kind of around that open core.” And we’ve moved towards, “Hey, we’re actually going to keep a lot of those core improvements. We’re going to keep those for the Datastax product.”

[00:34:17] JM: And you feel if you wouldn't have made that decision, you would be in a worse position today businesswise?

[00:34:24] JE: Yes.

[00:34:25] JM: Really. You say that despite the – I mean, because it sound like you have such heterogeneity of customers. Imagine if you would've just gone, like let’s say you would've just gone until like Intel today and just kept everything kind of in the open. Not keep as much stuff under the hood. What makes you sure that like you would be in a less strategically beneficial position?

[00:34:49] JE: I think what makes me sure is if you just look at the revenue numbers that Amazon Web Service posts, it is pretty much that simple, because if you're saying, "We have the same product or substantially the same product because we've open sourced it. On the one hand, Datastax is trying to convince people to pay for Datastax as service. And on the other hand, Amazon is saying, "Hey, we have all of the advantages in our cloud and we're competing with you directly by offering Cassandra as a service. That's a really tough proposition. And that's not a hypothetical situation.

We announced a couple months ago that we're creating a set of services called Datastax Constellation, and the first one of those is going to be Datastax Apache Cassandra as a service. Like I said, that's not a hypothetical for us.

[00:35:47] JM: Now, from a business point of view, I completely get it. Like so advantageous. What I wonder – And this is kind of like the bear case for the decisions that like Mongo and Elastic and so on are making here, is that there might be some downside from the point of view of the open source community.

But at the same time, it's like – I mean, how important is that open source community relative to the developers that actually work at the open core based company? Do you feel like there's been a tradeoff there?

[00:36:17] JE: Yeah. There's definitely a tradeoff. I'm not saying that this is a one-size-fits-all answer and everyone should do what Datastax does. It is a tradeoff, and I think – I said that 10 years ago there was kind of a sense that we understood how to monetize open source and build that. Now, we don't understand that anymore, and Datastax and Confluent and Elastic were all trying to figure out what the new answer looks like. There isn't a playbook where we're inventing new ones. So I'm not saying I've got all the answers, but I'm saying this is the problem we're trying to solve, and here's what I think is a reasonable first stab at a solution.

But to talk about the tradeoff there around the open source community, we definitely believe in the continued importance of that Apache Cassandra community. So when I said we're moving more of our innovations into the Datastax server product rather than contributing everything to Apache Cassandra. I definitely don't mean to imply that we took our ball and went home and said, "Good luck Apache."

So, we continue to maintain the half a dozen of the most popular drivers for Cassandra. We continue to be involved in meet ups or sponsoring Apache Con. Next month I'll be giving a talk about how Datastax is supporting the Cassandra 4.0 release with testing resources. I believe I may have another announcement to make. I'll let that be a surprise for Apache Con. We're definitely looking to stay involved with Apache Cassandra and continue to grow that success story.

[SPONSOR MESSAGE]

[00:38:11] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:40:00] JM: So we've been doing a lot of shows on this topic, and I just find it so fascinating, because people get emotional about this subject. And a lot of it has to do with what people view as fundamentals, open source fundamentals, like the ideological fundamentals of open source. When you think about it, like open source is not even that old and there was no like 10 Commandments of open source, right? Like people bickering the whole time about what open source mean.

[00:40:31] JE: Yeah. I think I could probably guess, like here're some personality traits that you find commonly in software engineers that might play into that. But, certainly, since Richard Stallman started the Free Software Foundation, I think it was in the early 80s. There's definitely that like, "Oh! No. It has to be GPL and it has enforce that people contribute back." Versus the Apache philosophy of, "It's open for everyone to use, and that includes proprietary uses." So, yeah, you're right. That's a fertile ground for debate for at least 30 years.

[00:41:13] JM: Tell me about the economics of a database company. I realize you're the CTO, and the organization humbly hired an external CEO, which is a decision I really respect by the way. Knowing when to hire an external CEO, that takes a lot of humility. So I have a lot of respect for that. But somebody smart enough to have the humility to do that, I imagine, has some sense of the economics of the business. So, tell me about the economics. Is there anything interesting you've learned about building a database business?

[00:41:43] JE: Yeah, I think this is probably relevant for the audience that you have here, which is when I started Datastax, I kind of had this mental model that we weren't going to need a sales force. Like, we'd have people to answer the phones when somebody called us up and said, "Hey, I want to buy Cassandra support." But the idea was the need for the product and the market demand was obvious enough that we wouldn't need a traditional sales force.

So, Matt Pfeil was my cofounder and our first CEO, and so he was the one who he actually put up his hand after about a year and said, “This is growing bigger than my ability to direct it and we need to bring in someone with more experience,” and that was Billy Bosworth, who became our CEO after a year. But when Billy came in, he’s like, “You guys need a salesforce like six months ago.” So he started fixing that as one of his first priorities.

But he was right. Like, we waited too long partly because as a couple people with engineering backgrounds, we didn't realized that when you go to a Cisco Systems and you say, “Hey, you should use our distributed database.” There's a whole process internally around making that decision and getting the CFO to agree to sign the check and so forth. So one of the things I would say I learned is that there is a lot more value to that sales side of things than I thought at first given my engineering background.

Other things about the economics. Today, Datastax is about 600 employees, and one of the things that I'm proudest of is that we’ve kept the culture of you can do your job from anywhere in the world. Obviously, there're some limits there around if you’re the sales manager for the Midwest, then you’re not going to be able to do that effectively from Singapore. But certainly on the engineering side of things, we have people from – Actually, it's about 75% in North and South America, about 25% in Europe and a handful in Asia, primarily because the Asian time zones are more challenging for US-based company. But, yeah.

I’d gotten used to kind of the open-source style of building software, and I really liked how the incentives just seem correct for engineering. In other words, if I'm running a team of engineers and they all come into the office and we have lunch together and so forth. There’s a very natural tendency to kind of manage by what you see. If I see Jonathan coming in and he's always in the office before I am and he leaves after I leave, I’m going to think, “Well, Jonathan really works hard.”

But the important thing is not in engineering how many hours you're putting in, but whether your work is effective in solving problems. So, when you're remote, it kind of takes that bias out of the system and now I'm managing by like what is Jonathan producing and is he doing reviews effectively and is he writing good code? Rather than this other half that's kind of a distraction.

I will say that that there is one additional skillset when you're working in a remote team like that, which is that you need to be better at handling things asynchronously. So, instead of going over to Gary's cube and saying, “Hey, Gary. I just posted a poll request for this problem. Can you check that out and we’ll get it committed?” Gary might be six hours away, and I'll have to wait until the next morning to see his comments and I need to be able to switch to working on something else in the meantime. But I think most engineers are okay with that, and I think it's a good tradeoff.

[00:46:01] JM: So if I know the timeline correctly, you started Datastax when you are in Austin, or was it San Antonio?

[00:46:09] JE: Yeah, I was in San Antonio and Matt Pfeil was in Austin. The company was incorporated in Austin.

[00:46:14] JM: Right. Okay. So, I guess you don't really have the MoPac traffic problem. Do you just work at your home or do you commute to an office?

[00:46:22] JE: Yeah. I commuted from San Antonio up to Austin for about a year a couple times a week just because our first engineering hires were in Austin.

[00:46:32] JM: Can’t do that today.

[00:46:34] JE: Yeah. Actually, we got office space in this very – I guess you could politely call it up-and-coming part of town on the east side of 35. It was very cheap, and that was good for us at the time. Today, that same spot, there's like three new hires.

[00:47:02] JM: Right. Right. Exactly. My mom was a real estate agent in ’08 in Austin around that time. So, like I drove around with her sometimes. Yeah, that was interesting time in the development of Austin. What a fast-growing city. But your headquarters are in Santa Clara.

[00:47:18] JE: Yes.

[00:47:19] JM: Why is it that all the database companies in the Bay Area are in like Santa Clara or around that area? Like all the Hadoop companies are in that area, like Rockset. I've interviewed them recently. They're kind of in that area. Is that like the best place to hire data people, or date data like just really like the microclimates in Santa Clara? What's going on there?

[00:47:41] JE: I think it's kind of – You probably know this better than I do, since you live out there, but there's kind of the old rule of thumb, is that San Francisco, that's more consumer-focused. Then Silicon Valley's, that's going to be more like enterprise-focused and also more hardware-focused. But that's less relevant for the database companies.

[00:48:02] JM: Interesting. So, you mention this like kind of competitive dynamic with the cloud providers. Could you tell me a little more about how that plays out in like deals or like how it plays out in terms of like tactics today. What do you feel is your relation to cloud providers? Is it a frenemy relationship and like how's the diplomacy look and so on?

[00:48:28] JE: The relationship with cloud providers is it's mixed. It is different depending on the cloud provider. For instance, Google announced a partnership with Datastax and a half a dozen other open source infrastructure companies at their conference in April. I think they're looking to be more proactive in kind of addressing that tension and saying, ",we want to work with you guys and create an opportunity for both of us to succeed," and they're looking at some creative ways to do that.

So, I hope that that's the wave of the future, because I really – We've got a great partnership with Google at this point and I'm really happy with the direction that that's going.

[00:49:15] JM: So, I've had some conversations with people about building these open source companies, and one point I've heard raised is that it seems to be important to be the moral authority of a project. So, like if you are building an open source company. It's kind of hard to do it if you are not the controller of the repository. Do you think that's strategically true or do you think that it's possible for multiple companies to be built around a single open source project?

[00:49:50] JE: You know, I'm not sure, because I can't think of an example of – Well, actually, yeah. I mean, the biggest example is Linux, right? So you've got Red Hat. You've got Canonical. So, yeah, I think that's definitely an example of multiple customer or multiple companies collaborating there. I think that people would have different opinions on whether each of these companies is pulling their weight so to speak in the project. But that's the best example I can think of. It definitely seems to be uncommon though.

[00:50:26] JM: How have you balanced engineering between the open source projects and kind of the internal sauce?

[00:50:34] JE: For the first six years or so, it was really easy where everything just goes to Apache. But the more interesting balance I think is more recently, like what does that look like now? It's kind of – Our VP of product up until recently, he moved into product marketing. His name is Robin Schumacher.

His rule of thumb was if you're building something that's a usual and customary feature of the product, like if you're building a database and you're looking at something like indexes. That's something that people expect to be in a database. That should be part of the open source project that you're giving away. But if you're looking at something like full disk encryption, that's maybe not as much of a usual and customary thing and that would be a good candidate for Datastax enterprise.

[00:51:29] JM: All right, last question. What's the biggest engineering problem you're working on today at Datastax?

[00:51:36] JE: That's definitely the launch of Constellation. So, we've been building this self-managed software for nine years now, and that of course that includes Apache Cassandra, as well as Datastax enterprise. And now we're saying, "Oh! We're also going to build and run a cloud service that exposes this to hundreds and thousands of users concurrently." That's a very different skillset and it's a very different muscle to grow internally.

So, it's both an engineering challenge in terms if you need to build the Kubernetes operator needed to build that Envoy proxies and so forth. But it's also an organizational challenge. So, that's – I would say that makes it more interesting when you have that combination.

[00:52:28] JM: Jonathan, thank you for coming on the show. It’s been a real pleasure talking to you.

[00:52:30] JE: Thanks so much, Jeff.

[END OF INTERVIEW]

[00:52:41] JM: When I was in college, I was always looking for people to start side projects with. I couldn’t find anybody. So, I ended up working on projects by myself. Then when I started working in the software industry, I started to look for people who I could start a business with. Once again, I couldn’t find anyone. So, I started a business myself, and that’s the podcast you’re listening to. But since then, I’ve found people to work with, on my hobbies, and in my business, and working with other people is much more rewarding than working alone. That’s why I started FindCollabs.

FindCollabs is a place to find collaborators and build projects. On findcollabs.com, you can create new projects or join projects that are already going. There are topic chat rooms where you can find people who are working in areas that you’re curious about, like cryptocurrencies, or React, or Kubernetes, or Vue.js, or whatever software topic you’re curious about.

We now have GitHub integration. So it’s easier than before to create a FindCollabs projects for your existing GitHub projects. If you’ve always wanted to work on side projects or you want to find collaborators for your side projects, check out FindCollabs. I’m on there every day and I’d love to see what you’re building. I’d also love if you check out what I’m building. Maybe you’d be interested in working on it with me.

Thanks for listening, and I hope you check out FindCollabs.

[END]