EPISODE 1305

[INTRODUCTION]

[00:00:00] JM: Welcome to Software Engineering Daily. The preamble today is recorded in my hotel room. I'm traveling, and I just want to get these preambles knocked out. If you want to fast-forward to a well-produced high-quality audio episode, you can just fast forward through this intro. It's the only part of the interview that's not on a good microphone. Why don't I have a good microphone right now? I've been very busy lately. And I expect this quality degradation to just be temporary. But I'd like to just get this episode out to you even though I don't have the microphone with me right now. Actually, more accurately, I don't have the correct USBC dongle conversion system. I left my conversion dongle at home, which is like kind of a sign of some sort of breakage in hardware, right? That I don't have my dongle with me. And there's not even a same day dongle delivery service. I was looking at Doordash and seeing if I could get a dongle so that I can plug in my USB audio interface so that I can use my high-quality mic that I brought. But I can't do any of this because I don't have a dongle.

So all that is to say, that's why I'm pursuing a couple companies that I've been spending a lot of time raising money for and thinking about. And I hope to post a lot about those ideas and those companies in the coming months. I'm finally having some success in raising money. And there's a lot that I want to share about company building and financial like fundraising stuff. I want to turn this into more of a podcast that explains how the software industry works from somebody who's trying to do software as an entrepreneur. So hopefully, that'll be an interesting direction that this takes.

Today's episode is about Pulsar. Pulsar is a cloud native distributed messaging and streaming platform was originally created at Yahoo. It's now a top-level Apache Software Foundation project. Pulsar is used by many large companies like Yahoo, Verizon, Tencent, and Splunk. DataStax is using Pulsar in an interesting way. And I'm, in today's episode, speaking with Jonathan Ellis, who is one of the most important contributors to the Apache Cassandra project. He's a founder of DataStax. And DataStax is widely known as the Cassandra company. But actually, they're doing something bigger. And there's a lot to that vision. I kind of got a sneak peek into that vision today, in today's episode, with Jonathan Ellis. And a lot of it has to do – If

you think about what is the core competency of Cassandra. Cassandra is a resilient, masterless distributed database, which is a pretty cool abstraction. And it's useful in a lot of ways. It's got a lot of performance problems that are hard to fix. Not Cassandra itself. I'm just talking about this domain of system that's being built at DataStax. And so there's an infinite problem depth here. And one of the questions is how do you do multi data center consistency? And that's not a question I was aware was an open question. But if you think about it, like, okay, we're getting these really good distributed systems. It's time to go multi data center in a lot of cases. If you can flip a switch and be multi data center resilient, that's great. And you should do that. How do you get that if your DataStax?

If your data stacks and your goal is to basically build a really good lock-free distributed database system, and you want that as your core competency, you've got to have some data center failover situation going on. And how do you actually make that a reality? Apparently, Pulsar is helpful. Why is Pulsar helpful? Because of Apache Bookkeeper. Apache Bookkeeper is basically a defined open source abstraction that does storage for you in a queueing system.

What is Apache Kafka? Apache Kafka is a resilient distributed queuing system that does not have a standalone abstraction known as Bookkeeper. So these are really two different systems. Pulsar is often compared to Kafka. They are like each other in some regard, but more accurately, they're just different systems. They have different tradeoffs. They have different focuses. They have different communities. And it's interesting to think about are there other open source projects out there like this that are being overlooked right now? Because Pulsar has been overlooked for a long time. It's got kind of an unusual history. So we've done episodes about Pulsar before. If you want to hear those, go back. But this one I think is actually really interesting, because I was convinced during the show that Pulsar was really interesting and was a useful technology, and I hope you have a similar revelation.

[INTERVIEW]

[00:04:43] JM: Jonathan, welcome to the show.

[00:04:45] JE: Thanks. It's great to be back.

[00:04:47] JM: Yes. Last time we talked about Cassandra. We talked about DataStax. And I think we could start there. There's so much depth to what you've done. So DataStax is the company that productized Cassandra. Cassandra was transformative for reasons that we explored in the previous episode. From a product standpoint, if you're building a company around a database, I see there's like one of two directions. So you can go either the MongoDB style direction where you take the core product, you orient the entire company around that product, and you say, “We're kind of going to build adjacencies to that product and try to upsell off of that core idea of a MongoDB database or a serverless MongoDB database. We're going to build like serverless stuff around that.” Or you can take kind of an Oracle approach and then you sort of expand into this whole product suite. And you really try to move beyond the essence of just being built around a database. And either approach can work super well. So what do you think about the difference between building your database into a platform versus building an entire company and think about the company as the platform?

[00:05:57] JE: I think as an engineer, it's a lot easier to approach things in the first way, because that's much closer to I'm building a product that I understand, that I understand the need for, that I would use myself, versus in the Oracle camp, you need to be not only an expert on databases and building applications on databases, but you need to be an expert on enterprise resource planning, and Salesforce automation, and so forth. So it's a much more difficult thing to do. And I think that's why you haven't really seen anyone do that since the 90s when Oracle really kind of pioneered that. I do think that that works then. I don't see anyone like trying to run that same playbook today.

[00:06:41] JM: Looking at DataStax, you have moved into some adjacencies to the initial core offering of Cassandra. How has the beachhead of what Cassandra offered you and the kinds of clientele that it offered to you? How has that led to adjacent product lines?

[00:07:00] JE: There's a couple adjacent product lines that we're moving into. So the first is we have kind of a core constituency of enterprise Java developers. Like there're lots of Cassandra clients. And you can use it from Python, or Go, or C++. But really, our core constituency is those enterprise Java developers. And one of the areas that we'd like to be a lot more present in is the Jamstack, the JavaScript developers who aren't necessarily interested in using kind of a classic

Cassandra CQL client. They just want to make a REST API call. They want to make a GraphQL call. And so we created a product called Stargate that is basically a proxy for Cassandra where we can translate not just those relatively straightforward REST API calls into CQL. But we've also started doing more advanced translations where we'll actually give you a schemaless API on top of Cassandra.

And so for your listeners who categorize Cassandra in the same general NoSQL category as MongoDB. Broadly speaking, yes, they're part of that same category. But traditionally, Cassandra has said, “You must issue a create table statement to tell me what your primary key is, and what kind of data is going to be in the columns in that table. And if you want to change that, then you need to do alter table.” So it's much more of a classic database experience for better and for worse. And so we've added this schemaless layer on top of that in this Stargate product, which is free and open source as well.

[00:08:47] JM: Well, you’re kind of blowing my mind right now, because I'm realizing that I think the reason MongoDB has more popular usage than Cassandra is probably because it was way easier to operationalize when the two are coming out. But Cassandra is, by definition, not easy to operationalize. But when you move both products into the cloud and basically offer them as a “serverless thing”. It kind of doesn't matter. If the operational differences don't matter at all, then it all comes down to how does it do in several different performance categories? So when it comes down to that, I don't think we did a head-to-head last time. Can you give me the head-to-head of MongoDB versus Cassandra? Condensed. Condensed.

[00:09:32] JE: Yeah. I’m a Cassandra expert. I'm not going to pretend to be a MongoDB expert. But what I will say is that I think your description of the two is accurate, that Cassandra definitely had a reputation that was deserved of being a lot more heavyweight and a lot more difficult to operate. And also, kind of that classic database developer experience that I mentioned, a lot of people aren't crazy about that either. So as we’ve our Astra Cassandra service, and Stargate is a part of that service. So you can use Astra in a schemaless fashion if that's what you want to do. So we launched AstraGA about a year and a half ago. So I think that was a little bit after we talked last. But a lot of what we've been doing since then has been motivated by people coming and using Astra and saying, “I really wish it were easier to do this.”

That kind of quick product feedback loop is one of the things that's so great about building a cloud service and just having that first party distance to the people who are using it.

[00:10:38] JM: Let's get to the impetus for this conversation. So you sent me an email where you said that you've been working with Pulsar. And this was a surprise to me. I've done a few shows on Pulsar. I've followed it. And my understanding of Pulsar was – Okay, so if you think about Kafka, Kafka was developed at LinkedIn at probably around the same time, like give or take two or three years, that Pulsar was developed at Yahoo, right? That's where Pulsar came from, I think. So you have Pulsar in Yahoo. You have Kafka in LinkedIn. And Kafka gets developed with slightly less legacy technology, because they're later than Yahoo, or in an environment of less legacy technology. It gets created kind of around Samza and these other things that LinkedIn is thinking about at the time. Pulsar came from a slightly different set of constraints and product ideas. For some reason or another, Kafka won out. I'm sure this is a story that is something like the MongoDB versus Cassandra thing where it's like, if you go back in time, there're certain reasons for this. Can you give me those reasons?

[00:11:42] JE: Yeah, just to update the timeline a little bit. Pulsar was released in 2016. So that was four or five years after Kafka came on the scene. And so I think that's a big reason for the difference in mindshare, is that there is that delta of a fairly large head start for Kafka. Can I give you the slightly long version of how I got to Pulsar?

[00:12:06] JE: Yeah, indulge me.

[00:12:07] JE: So one of the categories of enterprises that DataStax works with is fortune 1000 companies that are looking to modernize their infrastructure, and in particular, getting off of mainframe and getting off of Oracle client server. So they've got these monolithic applications. They're looking to get those cloud ready. I think that the hype around everything's going to be public cloud by 2025. I think that was a little oversold. But companies are definitely looking at this like, “Hey, I might want to run the next generation of this in a public cloud. And so I want to be ready for that. I don't want to have something that's super difficult to make that transition.”

So they're trying to decompose these monoliths into microservices. And so for a microservice, you've got the stateless application server tier, which is trivial to scale. So the interesting part is

what do you do with your state? What do you do with your data? And so you've got the system of record where Cassandra fits really well. And then you need a message bus to decouple those micro services from each other.

And so we've been looking at Kafka for four plus years as a complement to Cassandra in these use cases, and there's been one source of friction in particular, that's super difficult for us to get past, which is that Kafka is a single data center architecture. And one of the big reasons that people come to Cassandra is for our best in the industry multi data center replication. So those two, it's an awkward fit.

And so it's just kind of been on my radar of is there an alternative to Kafka that could be a better fit here? And I think that about 18 or so months ago, Pulsar got to the point in terms of stability and in terms of feature set where it solves the problems that people need in this micro service environment. And it also does what it says on the box without having to hire committers to dig in and figure out why you got this exception that you weren't expecting.

I was talking to a large Pulsar user recently. And he said, “Yeah, when I started using Pulsar, it was pretty solid. And today I would say it's bulletproof.” So there's an asterisk that I would put next to that that I can come back to in a minute. But really, I've been really, really delighted with what I found with Pulsar. I think the architecture is fundamentally better than what you get with Kafka. And I think that I would say it's better in general, and it's better specifically around I want to build cloud services on top of this thing.

[00:14:50] JM: So that's a profound observation, that Kafka is not built to be multi data center safe. What do you say consistent safe? Fault tolerant, basically? It might not sound like a big deal. But if you build your entire infrastructure around Kafka, like many people have, and you have a fire in your data center, like there was in AWS, what, like a few months ago? Wasn’t there a fire in an AWS center? It's literally on fire inside.

[00:15:17] JE: Yeah. It seems like about once a year, there's a big outage.

[00:15:19] JM: And there was OVH a while ago too, right?

[00:15:21] JE: Yeah. Oh, yeah, that was the fire. Yeah.

[00:15:24] JM: Yeah. So you definitely don't want to lose your company because of a fire.

[00:15:29] JE: There's a reliability aspect to the multi data center, but there's also a performance aspect, which is that if I have a message bus that only lives in one data center and I'm globally deployed with my application, then I'm having to eat a speed of light latency to that data center where the Kafka lives. Whereas with the Cassandra approach and the Pulsar approach is we're going to let you do locally synchronous reads and writes and then will replicate to other data centers asynchronously and do that in a manner that gives you consistency across all of those.

[00:16:06] JM: I mean, multi data center safety for your message bus. That's like national security level importance, because we're definitely at the point where like if one of our AWS regions has a nuclear bomb go off, like that's the end of civilization. I find it ironic that the internet was basically designed to be nuke safe. And we've actually made ourselves a single point of failure to nukes.

[00:16:32] JE: Yeah. And I remember, it was a couple years ago, but there was a big outage of all of US East. Not just one availability zone, but the whole region was effectively offline. And one of the big services that kept working during that outage was Netflix, because they're using Cassandra replication across multiple regions. So even losing all three availability zones, they were able to tolerate that.

[00:16:57] JM: Yeah. Anyway, so maybe you can motivate this for people. So first of all, you are still at DataStax, right? You're not going to start a Pulsar company, I assume, right?

[00:17:08] JE: I'm starting a Pulsar company inside DataStax.

[00:17:10] JM: Okay. All right. PulsarStax. Explain why Pulsar is critical infrastructure for DataStax.

[00:17:16] JE: Yeah. So this kind of comes back to your first question where you pointed out that database companies can verticalize and go after the markets for the applications built on top of the database. And we're trying to horizontalize here. We're going after this message bus component that we see as a critical complement to Cassandra. That people who are using Cassandra probably also need Pulsar and vice versa. I've talked to enough of our Cassandra customers that I think we validated this hypothesis. And now we're going through the process of actually turning those into Pulsar customers as well as Cassandra customers. So we launched our Pulsar as a service, open beta. We launched that last week. So you can definitely come check that out and see what that's going to look like when we go GA as soon as possible.

[00:18:10] JM: And if I get pulsar as a service, how is that going to compare API wise, SLA wise to Confluent Cloud, or Amazon Kinesis, or Amazon Kafka, whatever, Google pub/sub? What's your angle there?

[00:18:26] JE: I think there's three key requirements for modern cloud infrastructure for most of the enterprises out there. One is that it needs to be based on open source. That alone that prevents a whole category of problems that these companies have had with some of the traditional vendors in the space. So if you've got an open source piece of infrastructure, it keeps everyone honest. So DataStax can't go to our customers and say, “Hey, we're raising your price by a factor of 15 next year,” because they'll look at us and say, “Okay, well, I'm just going to go with open source Cassandra with open source Pulsar.” And it gives everyone that kind of insurance.

The next requirement is that it needs to be ready for hybrid clouds. So I mentioned that I don't think that we're moving towards a future where everything is public cloud. We're moving towards a future and towards a present reality where companies are taking the best of both of those options for deploying their infrastructure. So if you have a scenario where you have a highly bursty workload, you can take advantage of the elasticity of the public cloud. On the other hand, if you have an application where you're more concerned about security, or maybe you're more concerned about like I can get 50% better performance if I run this on hardware that I've selected exactly for this use case, then you can also do that and get cost savings for that use case.

Effectively, I don't want to say 100%, but it's pretty close to 100% of DataStax customers are doing hybrid cloud in some fashion. And so you need something that's open source. But you need something that, in addition to being open source, gives you the flexibility to run on your own infrastructure, on cloud infrastructure, and across those. So that's coming back to my first point about multi data center replication. If you have a piece of code that you can replicate from your own data center to the cloud, that allows you to do a migration all the way to the cloud without downtime.

Then the third factor that I think companies are looking for is that when they are adopting cloud, they want that to be with cloud native infrastructure. So that word gets used a lot. And I mean as something specific by that. I mean that it's not just a lift and shift of I'm running this same open source software, and I'm running that for you in the cloud. So you have your cluster with your nodes. And adding capacity means we're going to do some kind of addition of extra nodes to the cluster. And compare that to I have a multitenant cloud service, and I'm giving you a consumption priced model for that, where you're only paying for what you use, and you never have to think about how many nodes is this and so forth. So people are looking for that second one. They're not just looking for Apache Cassandra in the cloud or Apache Pulsar in the cloud. They're looking for a multitenant elastic service that they can burst up and down without having to deal with actually moving nodes in and out of the cluster.

[00:21:41] JM: That's an amazing vision, because you're basically saying people don't want raw EC2 instances. You don't want to be using EC2 instances. You want some kind of Heroku-like experience. You want all that annoying plumbing to be taken care of. So if you're trying to build this kind of service, you got Pulsar. I get what you're saying. I get we're trying to build. You've got Pulsar. And help me understand the open source side here. Do you have to adapt and modify and fork the code to have the DataStax specific support that you need? Or like architecturally, do you just build shims around it into the DataStax specific architecture? Like what's the strategy there?

[00:22:20] JE: So one of the things that was super attractive about Pulsar is that it has almost everything that you'd look for if you were saying, “How do I build a multitenant cloud service on top of this?” So it has multi tenancy that's built-in. It has separate compute and storage components. So you can scale the brokers independently from the Bookkeeper storage nodes

and vice versa. Having that separation of compute and storage also means that you get tiered storage as a first-class citizen in the architecture. So I'm going to rat hole on that for just a minute. Because one of the advantages of using a modern message bus like Pulsar, it's storing those messages durably on disk. So it's not like last year's message queue where everything's in-memory, and then once it's delivered, then it's gone.

So with Pulsar, you can replay events to test for a bug that you think you fixed, but you need to replay that to run the corrected logic. Or you can capture that and use that as part of your testing cluster that you can rerun that traffic from your live cluster, probably suitably anonymized, and so forth.

But a lot of those advantages, you lose those if you say, “I'm only going to retain this data for a week, because that's all the space I have on my local disks.” So Kafka and Pulsar, like that's their primary way of being deployed is I've got all my data on the node that's responsible for that. But as part of the separation of storage out into the Bookkeeper layer, Pulsar also lets you store data into things like S3, into things like HDFS, and still access that with the Pulsar API's. So you've basically got a hot storage locally, but then you've got warm storage that you can expand into and say like, “Hey, I'm probably not going to use this data every hour moving forward, but I do want to have access to it down the road and so I can kind of archive that off to another system.” That's a very powerful cost of ownership thing, because I'm just having to size my Pulsar cluster for the throughput and not having to size it for the cold storage. I can do a million requests per second on a three node pulsar cluster. That's a legitimate thing that people can expect to achieve. That's not black magic.

But where people have to start getting into dozens or hundreds of nodes is, well, I want to retain a year's worth of data. And I just need that much nodes to store all of that on. But now with Pulsar, I can just archive that off to S3. And so I'm just paying for that three nodes that's handling my hot data throughput.

So there’s the multi tenancy. There's the separate compute and storage. There's the multi region replication. And then the one more thing that's just kind of an extra icing on the cake is that when you expand a Pulsar cluster, expanding it doesn't make it slower before it makes it faster. So what I mean by that is, in Kafka, because the storage is tightly coupled to each node

in the cluster, if I want to expand that cluster, I want to add a new broker node. I need to move data. I need to move some of those partitions to the new node. And so I'm imposing this additional cost on the existing nodes of reading that data off-disk and sending it over the network to the new node. Then once that's been done, now the new node is serving that data and I've increased the capacity of my cluster. But while that's happening, while it's moving that data over to the new one, it's making my clusters slower. And so obviously, that's a tricky place to be in if you're like, “Oh, crap, I need to expand my cluster,” but expanding it is going to make it slower until that finishes.

So in the Pulsar world, they've introduced an extra layer of abstraction where the topics and the partitions are broken up into a Bookkeeper unit of storage called a ledger. And ledgers are striped across different nodes in the Bookkeeper storage ensemble. And so this actually gives you two things. One is that a single partition, because I'm striping it across lots of storage nodes, I can actually get a lot higher throughput in that single partition before I have to break it up into multiple partitions. And so that gives me some nice to have things around the guarantees around message ordering are per partition. And so once you have to break that up, then you have to think harder about how your application is going to use that. So if I can get by with a single partition, I'm going to be happier as an application developer as an architect.

But then the other thing that you get from that abstraction layer is, when I add a new storage node to that Bookkeeper cluster, all I need to do is start allocating new ledgers that include that new storage node. I don't have to move any of the existing data.

[00:27:35] JM: I think what you've mentioned here is Bookkeeper, the storage system that underlies Pulsar. Is there an analogue in Kafka? Kafka is just like using storage. You just like plug in storage, right? There's not like a dedicated abstraction for storage in the pub/sub system, right?

[00:27:55] JE: Exactly. Yeah. In the Kafka world, every broker manages its own local storage.

[00:28:00] JM: Okay, every broker manages its own local storage as opposed to having a defined storage orchestration layer.

[00:28:08] JE: Yes. In Pulsar, the brokers are actually stateless. So all they're doing is basically – Like all they're doing, like it's more complicated than that. But basically, they're just taking Pulsar API calls and translating that into Bookkeeper storage requests. So you can scale both of those halves independently. You can add more Pulsar brokers without adding more bookies. Or you can add more bookies if you need more throughput on the storage side.

So you mentioned, how does this compare to cloud, pub/sub on the Google side, to Amazon Kinesis, or to Confluent? We're finalizing our pricing on Astra streaming, but I'm confident that we're going to be able to beat any of those in part because of the efficiencies that you get from having that specialization across the components.

[00:28:57] JM: That doesn't surprise me. Actually now that you're describing it, Pulsar seems like a different product category than Kafka. Kafka is more of a platform. I think when you're using Kafka, what you really are getting is this nice ecosystem of plugins and KSQL and just basically sugary things on top of it that are really helpful from an application developer standpoint. But you're telling me the lower level plumbing is different in a way that may hurt you in some situations.

[00:29:27] JE: Yeah, I'm preaching the gospel of Pulsar on two levels. First, I'm claiming that Pulsar is a better Kafka and Kafka. And we can double click on what kind of compatibility story there is there. But I'm also claiming that Pulsar can do things that Kafka can't because of these architectural limitations. So let me bring that up to just a little bit of a higher level to talk about what the implications that has for application developers.

So Kafka is kind of a classic pub/sub implementation, that you have topics, you can publish to those topics, you can subscribe to those topics. And every subscriber gets their own copy of the messages that were published there. The other thing that people talk about in the streaming space is queueing. So this is where if you're thinking about JMS, or you're thinking about AMQP, you're thinking about where people send messages to a queue. And then those messages are delivered in a FIFO order to consumers, but they're only delivered to one consumer. It's basically load balancing across the consumers.

And one of the reasons that Kafka can do the former, it can do pub/sub, but it can't do queueing, is that it's designed around this batch architecture, where each consumer says, “Here's my high watermark of the messages that I've seen.” And so you acknowledge messages in bulk by advancing that offset. And in the Pulsar space, this is actually one of the things that the engineers at Yahoo explicitly looked at and said, “We want to improve on what Kafka does,” is they designed it so that you can bulk acknowledge that way you can in Kafka, but you **[inaudible 00:31:17].**

And so what that means is now I can fan-out those messages across consumers and load balance them the way I would for an AMQP implementation. And in Pulsar’s terminology, that's called a shared subscription. And so now, if one of those messages isn't acknowledged, I can replay that message to a different consumer and I don't have to replay a thousand messages, because I'm doing individual acknowledgement. And I can do that because Pulsar’s architecture gives me the flexibility to do it.

And so we've actually taken that from a theoretical thought experiment and turned it into a product. So we created something called Fast JMS, which is a JMS implementation on top of Pulsar, and it is TCK compliant. So it passes the tests. It is a real JMS implementation, but –

[00:32:10] JM: Can you remind me what JMS is? Java message bus or Java message system?

[00:32:16] JE: Yeah, that’s right. Java message system. Now it’s the Jakarta message system. So you have like ActiveMQ. You have IBM MQ. You've got TIBCO that are going to give you JMS implementations. So it basically creates – It's a bunch of interfaces that say this is how Java applications can send messages to this system and your read them. And so we can implement, we have implemented JMS on top of Pulsar, and that gives you not just the scalability of a distributed system, but it gives you those other benefits of Pulsar. It gives you the multitenancy. It gives you the tiered storage. It gives you the geo replication. And you get all of those basically for free, because it's built on Pulsar.

[00:33:00] JM: So JMS is really cool, because JMS basically decouples your Java application from the specific implementation details of message system. So you may want one message system at one stage of your application and want a different message system at a later stage of

your application. So that's cool. Out of curiosity, if I want to have that kind of dependency agnostic interface in, let's say, the JavaScript world, is there an equivalent API thing?

[00:33:30] JE: I'm not aware of one, but I'm not an expert on the JavaScript world. So I'll have to –

[00:33:34] JM: Man, somebody should make that. That'd be great. I'd love to have that. I'm sure there's like Node.js API's into Kafka or whatever.

[00:33:43] JE: Well, actually, let me give a slightly different answer there. The answer earlier is correct as far as it goes. But I think we're moving towards a world where the Kafka API is actually turning into that lingua franca.

[00:33:57] JM: Oh wow!

[00:33:58] JE: Yeah. So you've got Kafka itself. You've got explicit Kafka clones, like I think one of them is called Red Panda. And so then you've got things like Pulsar that they're saying, "Hey, we're not Kafka, but we speak to Kafka API."

[00:34:14] JM: Wow! Okay, so Red Panda actually has a place then? Because when I interviewed them, I was baffled at what the heck this thing was or why it was being worked on. But it seems like it makes sense, if it's that easy to switch over to at least.

[00:34:25] JE: Yeah. If you're looking at Kafka, as you mentioned, like there's an ecosystem of connectors and so forth. And if you look at that as, "Oh, as long as I speak the Kafka protocol itself, I can use all of this ecosystem," and whether it's like Apache Kafka itself, or its Apache Pulsar, or it's something else, that's kind of an implementation detail.

And so with what the Pulsar project has done is they abstracted its processing of the network protocol and they basically created what they call a protocol handler plugin interface. And so you've got the native Pulsar protocol, but you've also got an implementation for Kafka. And so you can use things like Kafka streams on top of Pulsar using that protocol handler.

[00:35:14] JM: Okay.

[00:35:15] JE: I'll mentioned one more thing on the Kafka compatibility side, which is, in particular around all those connectors, which is by far the most common way to use Kafka, most people aren't writing code to the Java API or the JavaScript API. Most people are saying, "Oh, I want to connect Cassandra with Elasticsearch. And so I'll take the Cassandra source and connect it to the Elasticsearch sync." And somebody has done all the effort to send those messages to and from Kafka. And so one of the things that DataStax has contributed to Apache Pulsar for the 2.8 release, which came out last week as well, is the ability to use Kafka connectors inside Pulsar. So you don't need to run Kafka connect itself. You can just use that library for connecting to Cassandra for connecting to Elastic, you can use that inside Pulsar.

[00:36:10] JM: We're talking at a pretty high-level here. And I wonder, from your point of view, from somebody who has been watching distributed systems since they were coming into their own, how much has Kubernetes helped with the development process of this kind of stuff that we're talking about?

[00:36:32] JE: Yeah. So first of all, I'm going to –

[00:36:34] JM: I hope you tell me it's overhyped. Is it overhyped?

[00:36:36] JE: I'm not a Kubernetes fanboy, in the sense that I think it's inelegant in a lot of ways. But it's one. Like there isn't a reasonable alternative that you can tell people –

[00:36:48] JM: This is Heterodox. This is the Heterodox infrastructure podcast today.

[00:36:52] JE: So I've made my peace with Kubernetes. And in fact, our cloud services are built on Kubernetes with the eye towards if somebody wants to run Astra Cassandra or Astra Pulsar on their own VMware, or Red Hat, Kubernetes installation, then we can give them the operators, the control plane to do that.

[00:37:13] JM: So if Kubernetes didn't win because of superior technology, why did Kubernetes win?

[00:37:17] JE: Well, I mean, it's backed by Google. And there's a critical mass past which you're kind of spitting into the wind, right? And I think we hit that point a couple years ago.

[00:37:26] JM: But wasn't it at least the least bad architecture? Like was Mesos significantly better? Was Cloud Foundry better?

[00:37:33] JE: You could be right. I'm not looking to argue the details of how good the architecture is.

[00:37:40] JM: Alright, fair enough. But I'm sorry. So I got a little sidetracked though. But does it help to at least have a standardization? At least we have – Maybe the Linux distribution we've built sucks, but at least we're all using it.

[00:37:50] JE: Yeah, I'd have to think about that analogy to see how much I agree with it. But yes, it does help a great deal to have that common layer that you can start building against.

[00:37:59] JM: Okay. So can you sell people Pulsar? Do they care? Or has Kafka won that spitting into the wind thing?

[00:38:07] JE: So yes, and yes and no, I think in that order. So yes, we're selling Pulsar to people in two ways. We're selling the cloud service that I mentioned. And we're also selling a Luna Streaming support and distribution for people who want to run Pulsar themselves. And so what people don't like is rewriting code, right? So if I were going there and saying, “Hey, I want you to rewrite everything to use Pulsar.” I think that would be super difficult despite the technical advantages that Pulsar has architecturally. But that's not what I'm telling people. I'm telling people, “Hey, I have a JMS implementation that you can just start using tomorrow.” I'm saying I have a Pulsar cluster that speaks the Kafka API. And so I'm telling my messages, you don't have to rewrite your code. You can move it to Pulsar without doing that rewrite. And so yeah, people are very interested in that.

[00:39:06] JM: What's the process of getting to a point where you feel comfortable handing this over two people as critical infrastructure?

[00:39:14] JE: So I'll back up a little bit to one of the first things I said here, where I said that Pulsar is bulletproof. And I said that there's an asterisk next to that. So the asterisk is that Pulsar is still at the kind of adolescent stage of open source projects, where it's adding new features faster than it can keep those stable. And what I mean by that is – So I mentioned we have distribution of Pulsar called a Luna Streaming. The version of Luna Streaming that's out right now is based on Pulsar 2.62. We're working on a version of Luna Streaming based on pulsar 2.72. That's going to be out in the next couple weeks.

Meanwhile, Apache Pulsar released version 2.8.0 last week, and it's going to be several months before we start looking seriously about pulling that into a Luna Streaming, just because there's that dot zero effect of it's not quite production ready. So my advice to anyone who's looking at exploring Pulsar is, if you want something that's rock solid and it just works, we've kind of done that homework for you with the Luna Streaming distribution, which is, by the way, that is free and open source. So there's nothing proprietary there. So you can just grab that and run it. You don't have to talk to a salesperson or anything. But if you do want to use the latest and greatest features, by all means, try out 2.8.0, but try it out in a testing environment. Don't just throw it in production and hope things are going to work, because there's probably going to be some sharp edges.

[00:40:50] JM: Hey, why didn't something Erlang-based win the messaging bus wars?

[00:40:55] JE: RabbitMQ is Erlang based, right?

[00:40:57] JM: Okay.

[00:40:58] JE: So there's definitely a presence there. But I think the JVM is just more performant and that that counts at scale.

[00:41:06] JM: Got it? Okay, so the people who want a message bus from DataStax, is the primary motivation, the primary selling point, basically, “Look, we're going to give you a message bus with multi datacenter replication.” Is that the selling point?

[00:41:19] JE: I think there're a couple selling points. So I'm starting with DataStax as existing Cassandra customer base. So that's something that they tend to care a great deal about. But I'm also going to IBM MQ users. I'm going to Kafka users. And my message to them is, "I'm going to save you money." That's something that people like to hear. And so with the tiered storage, with the multi tenancy, that's my broader message for people who maybe aren't as interested in the Cassandra part.

[00:41:47] JM: What kind of cost savings do you see you get? Like 80% or something?

[00:41:50] JE: Yeah. So we worked with Giga to do a third-party evaluation. And they said you can save 81% with Pulsar over Kafka.

[00:42:00] JM: 81%, ha. But Kafka must be doing something about this. I mean, especially now the AWS is involved, they have their own Kafka product. They must be trying to get multi datacenter replication, right?

[00:42:11] JE: So there are add-ons to Kafka that will do multi data center replication. The most popular one is called Mirror Maker. But it's kind of the classic problem when you're trying to do something that's very core to a service that it wasn't designed to do. And so we saw this 12 years ago with MySQL, where DataStax's first Cassandra customers were all getting off of MySQL, or sometimes Oracle. And I remember one in particular, they were moving off of Oracle and Golden Gate replication. And they say, "Well, we've gone from a replication delay on Oracle measured in hours to a replication delay on Cassandra measured in seconds." And so, nominally, yeah, you can say yeah, Oracle does multi data center replication, but it's more fragile. It's less performant because it wasn't designed to do that. And that's kind of the situation that Kafka is in there.

[00:43:06] JM: And in making it multi datacenter replicable, I guess they might make it even more expensive, right? Like it's probably going to add – Well, I could be wrong about that. I mean, does it necessarily add overhead if they're bolting on? Well, I don't know.

[00:43:17] JE: Yeah. That's not something that I'd hammer on, because when you're saying I'm replicating to multiple data centers, you're necessarily saying I'm going to double my infrastructure footprint because I'm sending that somewhere else as well.

[00:43:30] JM: Yeah. Okay. Fair enough. Okay, very interesting conversation. You have really turned my head on Pulsar. What else has been on your mind lately? You're Jonathan Ellis, I'm sure you got some expansive thoughts on infrastructure that you've been brewing on that have nothing to do with Pulsar.

[00:43:48] JE: Can I give you one that actually does have to do with Pulsar, but –

[00:43:51] JM: Okay, fine. Alright, sure.

[00:43:52] JE: Which is that I think that a lot more people who, if you're building infrastructure, like if you're interested in creating systems like Cassandra or like Pulsar, I think that Bookkeeper is a secret weapon that's significantly underappreciated in the industry for what it can do. Basically, it gives you a log structured, replicated storage mechanism that you can apply to all kinds of problems. So besides Pulsar, Salesforce uses it as part of their internal database for application storage. So that's scaling to hundreds of terabytes. There's a system called Pravega coming out of Dell EMC. It's basically competitor to Kafka and Pulsar. Also uses Bookkeeper, but otherwise completely different from Pulsar.

ByteDance, which is the company behind TikTok, they use Bookkeeper for their internal metadata storage. And Twitter has a NoSQL database called Manhattan that also uses Bookkeeper for it's right ahead log. So big fan of Bookkeeper in particular in terms of distributed systems that do one thing and do it well. I mean, if you look at Cassandra, for instance, Cassandra isn't really designed to be torn apart and say, "Look, I'm going to use the gossip system from Cassandra to do something else. Or I'm going to use Cassandra's storage layer to do something else." It's not designed for that. But Bookkeeper is designed to be used in a pluggable fashion like that and more people should know about it.

[00:45:29] JM: Okay. Well, Jonathan, thank you so much for coming on the show.

[00:45:32] JE: Thanks so much for having me again.

[END]