**EPISODE 1306**

[INTRODUCTION]

**[00:00:00] JM:** Happy birthday to me. I'm recording this on my 33rd birthday. And it's been a great birthday. One event in today's birthday, I got to pitch Andreessen Horowitz's crypto team, notably Angela Strange, on Rectangle, this open source payment system that Yad Conrad and I are working on. And it was kind of a dream come true because I basically modeled my media company off of the same ideas as Andreessen Horowitz. And I finally got to pitch them with Yad alongside me, and it was just very fun. And I felt like they took the idea seriously. I'm not sure if they'll invest yet. I hope they do. But in any case, it was definitely like a highlight and a really nice 33rd birthday present. Some other stuff about that business will be coming soon. We're looking for a really, really good payments CTO. So if you know somebody or if you are a really, really good payments CTO, then we'd love to hear from you.

Today's episode is also about Pulsar. We did yesterday's episode about pulsar, and today's episode is going to be a pulsar also. Basically, Jonathan Ellis blew my mind too much, and I had to do another show with somebody from DataStax. I think he's from DataStax, right? He was either not from DataStax or he was from DataStax. Anyway, Enrico Olivelli is a distributed systems expert. He's one of these people that's been on distributed systems for a long time since applied distribute systems was a things, specifically data systems, data engineering, before it was called data engineering. And he is here to explain even further why Pulsar is interesting.

[INTERVIEW]

**[00:01:49] JM:** Enrico, welcome to the show.

**[00:01:50] EO:** Hello.

**[00:01:51] JM:** It's really great to have you here. I had a conversation just a few days ago with Jonathan Ellis, who is one of the founders of DataStax, which is the Cassandra company. And Cassandra is very interesting. A distributed database that's been around for a while, and the

open source project has grown in parallel with the company DataStax. Now, as DataStax has grown, it's had a few crossroads in product development. And the direction that you've gone recently with Pulsar is strategically interesting, because I can't think of kind of a strategic movement of an infrastructure SaaS company that is similar to this. So in order to explain the novelty to the listenership, let's just go right into – I think people who don't know Cassandra or who don't know DataStax can look back into our old episodes. But let's just fast forward to assuming people know what Cassandra is. They know what DataStax is. Why is Apache Pulsar relevant to your company?

**[00:02:51] EO:** Yes, because Apache Pulsar allows you to move your data next to where you want to store and where you want to process your data. So okay, you can start data in your Cassandra, in your cloud service, but you have to connect it with all of the other systems in your enterprise or connect with your third-party systems. And with Pulsar, you can achieve easily this because it is scalable. It was designed with multi-tenancy in mind, with security, with geo-replication. And also, it comes with a lot of connectors that are already out of the box. So it is really easy to connect your system with Cassandra. And you can scale. You can move your data whatever you want. You can interact with Internet of Things, systems.

**[00:03:48] JM:** And just to retread the conversation that I had with Jonathan Ellis a few days ago. It's really important that we convince people here that Apache Kafka and Apache Pulsar are different categories of technology. This was a mistake that I made in understanding the technologies. Explain why Kafka and Pulsar are actually in different categories.

**[00:04:14] EO:** Sure. I was between the early adopters of Kafka when LinkedIn donated it to Apache Software Foundation. So I've been following the old story. From the beginning, Kafka was something very useful to connect loosely decoupled systems, but it is only a very simple bus for data. Pulsar is a new generation. And when it started in Yahoo, and I know many people that work at the time in Pulsar, they started from scratch, because they needed geo-replication. They needed multi-tenancy. Kafka is something that is good to store your stream of data. There is a limited set of features regarding retention or scalability. And Pulsar started from scratch with the idea that it must be easy to scale, scale out to distribute your data all over the world. So it's another kind of product really.

**[00:05:21] JM:** You are the PMC of BookKeeper, right. So PMC, is that primary committer? What does that stand for?

**[00:05:30] EO:** No. It's PMC member. And the PMC, it's a committee that is driving the project. Because in Apache Software Foundation we have thousands of projects, every project has a committee that is responsible for the community and for all of the legal stuff, because in Apache we deal with open source software. So you have many people that use a software, many people that contribute their patches. You know, the name Apache comes from patch, from the word patch.

**[00:06:03] JM:** Apache Web Server.

**[00:06:04] EO:** Yeah, that's it. And so many people contribute to the project. And some of them are elected as committers. And basically they have write access to the repository so they can accept patches. But you need the PMC. The PMC is a bunch of people that decide when the project is ready to cut a new release. They verify that the project is moving forward in a healthy way. So basically, for instance, if I contribute a patch to an Apache project, I must be sure that I'm contributing it in the spirit and under the effects of the Apache license. So the PMC must guarantee this. And when you cut a release for a software, the PMC must vote for the release.

The other thing, there are tasks for the PMC, is to invite new committers. And so it is the role of selecting contributors and understanding that a contributor is able to continue the project to give time for the project. And the thing that I want to note is that in Apache, everybody works as an individual, not as a company. So I'm participating in many projects as myself, not on behalf of my company.

**[00:07:29] JM:** Alright. Well, BookKeeper is, from what I can tell, critical to the story here. Because in Kafka, if I remember what Jonathan told me correctly, in Kafka you don't have a storage abstraction, really .You have storage. And you can do stuff with that storage. And you have in-memory. And you can kind of do stuff with that in-memory. But BookKeeper is a dedicated striped storage system that underpins Pulsar. Do I understand that correctly?

**[00:08:01] EO:** Yes, that's. It is true. And we can also think about Pulsar as a higher level on top of BookKeeper, because in BookKeeper you have these ledgers that are immutable streams of data. But you can open this ledger and you can write it once and then BookKeeper stores it in a scalable way, but it is immutable. So once you stop writing to a ledger, you want to start again. You cannot open in a pen mode ledger. You have to start a new ledger. And also only one machine can write to a ledger at a time. So basically, you open one ledger, you write, and you close. And Pulsar adds all of those obstruction and implementation that are needed to concatenate BookKeeper ledgers as an infinite stream of data. BookKeeper is the right choice, because BookKeeper was born as a writer log for databases and for HDFS. And so it is really fast for write. But during the years, the community grew and added many features that helped BookKeeper to be ready to support catch-up reads. So BookKeeper was very fast in writes, very fast in tailing reads, that is to read the things that you have written, but it is also fast in reading all the data.

**[00:09:42] JM:** Can you explain the term striping to me in this context?

**[00:09:47] EO:** Let me make a little compilation with Kafka. In Kafka, you write on the broker, and all of the data must reside on the broker that is accepting the write. And you can replicate on other machines. But the full stream or topic must reside on the broker. In BookKeeper, for instance, if you want to write with the replication five, so if you want to write three copies for each message and you have a five BookKeeper servers, bookKeeper will send three copies of the message in a way that the messages are spread among all of the machines. So basically, you can add more machines, and the system will automatically scale out and spread all of the messages among all of the machines automatically. It's a great feature of Pulsar, and it's a power of BookKeeper.

**[00:10:43] JM:** So I think it's worth talking a little bit here about your work history, or your open source history I should say. So you've done work on Maven, ZooKeeper and BookKeeper. I want to talk about all those. Let's go to ZooKeeper. So ZooKeeper, the distributed lock server based off of the Google Chubby paper, right?

**[00:11:05] EO:** ZooKeeper, it's like a simple key value store. You can think it more about a little file system. And it was born in Yahoo more than 10 years ago, probably. But the idea between

ZooKeeper was to have a very simple system to store configuration and to allow coordination to have only a single system that deals with the consistency of the overall distributed system. Because at Yahoo, at the time, they had many services that add to share configuration or share leadership, execute the code that must not be executed in parallel, concurrently, critical sections. And most of the times, when you start a new application, you want distributed application, you dare to write everything from scratch. But it's not easy to implement such a coordination system.

So when ZooKeeper was born, they started with this very simple system that looked like a file system. So every developer could learn about it pretty quickly. And that is a foundation for many other systems. Actually, it's the sub-protocol. But the power of ZooKeeper is that it is simple. And the principle, keep it simple. Do only one thing, but do it very well. I see that in the ecosystem a few projects wants to get rid of ZooKeeper. But if you are using ZooKeeper for what ZooKeeper is doing, ZooKeeper is great, and it is running at scale. And it is the core of Facebook, for instance, and Twitter. So there are companies that are running ZooKeeper at very large scales.

But if you ask ZooKeeper to do something that is not for ZooKeeper, then you are wrong. For instance, if you want to write very quickly, you should use something like BookKeeper. BookKeeper leverages ZooKeeper for metadata for dealing with consistency of the data. But BookKeeper is able to do well its job. That is to be very fast.

**[00:13:24] JM:** Sorry. I may be stupid here. But what is the critical infrastructure at Facebook that is underpinned by ZooKeeper?

**[00:13:33] EO:** Probably I can't tell you much, because I know something, but –

**[00:13:35] JM:** Oh, Cassandra. Cassandra?

**[00:13:37] EO:** No, Cassandra is not using –

**[00:13:38] JM:** That's right, because Cassandra is masterless.

**[00:13:40] EO:** Yes, yes, yes, it is not.

**[00:13:43] JM:** Okay. So it's not Cassandra. So it must be Kafka?

**[v] EO:** No, no, no, they're using it. Probably I can't tell you pretty much. But you take about configuration if you want to share configuration among millions of machines.

**[00:13:58] JM:** I'm glad to bring this up, because I don't know if you know. I recently wrote a book on Facebook. So I spent two and a half years writing a book about how Facebook builds software. It's called *Move Fast*. And you're uncovering the real benefit to writing it, which is I've basically written a glowing review of how Facebook builds software. Like a glowing review. It's just such positivity about their software strategy, and I sincerely genuinely mean it. And now I'm really hoping that they take me on a tour of the company. And they say, "Okay, here's what we didn't talk about."

**[00:14:27] EO:** Yeah. But I can share with you that during the past two years, Facebook wanted to contribute back all of the improvements to ZooKeeper. Their project name is **[inaudible 00:14:38]**. And they started two years ago. Initiative, called ZooKeeper friends at Facebook, and they run a few meet-ups, and they contributed many, many, many features back to – Because they forked ZooKeeper a long time ago. And they contributed back lots of features. So everybody now can enjoy the feature contributed by Facebook. But probably we are not making such advertising in ZooKeeper, because the community is very little. Only engineers and engineers do not like to write docs, or to write logs and talk about things. So it's a pity.

**[00:15:17] JM:** Can you tell me about what were the bugs that Facebook smashed out of ZooKeeper recently?

**[00:15:23] EO:** I don't know really.

**[00:15:25] JM:** Well, okay. You don't know about the improvements or anything?

**[00:15:28] EO:** Oh, improvement? Yes. Yes. Yes. Bugs? I don't know.

**[00:15:31] JM:** Oh, okay. Oh, I guess – Okay. ZooKeeper is probably pretty bulletproof after all these years bug wise. So what's the difference between removing a bug and adding a feature in ZooKeeper? Is it latency? Are those improving latency?

**[00:15:41] EO:** Oh, yes. For instance, they contribute a lot of feature about security, allowing people to switch smoothly to TLS, for instance, or to create complex hierarchies. In ZooKeeper, you have observer nodes. So you can create very complex hierarchies of clusters of ZooKeeper. And this is contributed by Facebook. Also, they contributed a lot of stuff about observability metrics. I collaborated with them with the new metric system. One step at a time, they could do many things that allow you to run ZooKeeper, again, at scale and at Facebook's scale.

**[00:16:24] JM:** If we imagine the ideal metric system for ZooKeeper, what are the KPIs of that metric system?

**[00:16:31] EO:** Oh, everybody in ZooKeeper cares about availability, because usually if ZooKeeper is not available, nothing works. And latency, because usually in ZooKeeper you use it for locks, or to share configuration updates. So it is important that you're very fast in writes. But also the notifications of the changes are very fast. So I would bet on those two metrics, availability and latency.

**[00:17:04] JM:** What's the typical ratio of reads to writes on ZooKeeper? Is it even?

**[00:17:08] EO:** It depends on your system. And I believe that probably it is better that you read more often than write. But it's not always the case. I know I've worked in systems that use ZooKeeper mostly for distributed locks. So for a distributed lock to write, to acquire a lock. So it's mostly write. It depends on your case.

**[00:17:33] JM:** I always forget this. And it always makes me feel dumb when I talk about Cassandra, but I can't remember how the masterless system works. I think it has something to do with gossip. But can you remind me how Cassandra works without having a lock server?

**[00:17:47] EO:** I really sorry, but I don't know much about Cassandra really.

**[00:17:50] JM:** Oh, you don't know either. Oh, good. That's hilarious. Wait, do you work for DataStax? Or did I get that wrong?

**[00:17:56] EO:** Oh, yes. I know that it's gossip behind the scenes. But really, I'm a Pulsar guy in DataStax.

**[00:18:04] JM:** Oh! So you do work in DataStax.

**[00:18:06] EO:** Yes, I work in DataStax, in the uniStream team. And we're working in Pulsar. And we're connecting Pulsar with Cassandra. Also, I have written an enterprise level connector between Pulsar and Cassandra. And I'm working also in a new series of connectors for captured data change. But yes, I'm not a Cassandra expert right now.

**[00:18:31] JM:** So if we're talking about what makes Cassandra special. So I know it's a distributed database. I know it's masterless. I know I can use it in similar contexts, so when I would use MongoDB or, I presume, DynamoDB. And my understanding is that one thing that's difficult to do is – Please inform me if I'm wrong about this. But it's just I've been thinking about my conversation with Jonathan. I was like trying to understand like what's going on here. So I think what you have is you have Cassandra, and Cassandra is good in a single data center context. But if you want to have multi-data center, Cassandra, you want some way to pass messages between the two data centers. Pulsar is how you do that.

**[00:19:19] EO:** Really, I'm not an expert on Cassandra. But since I've started working in DataStax, I saw many amazing things about Cassandra in multi-region, multi-data center without Pulsar. So even I'm not an expert of Cassandra, I'm sure that there are many great things that you can do, because, really, now that I work with DataStax, I heard about many interesting stories about this. And I'm also following from a distance many work that my colleagues are doing on AstraDB, and severaless, and multi-region databases. So I believe that Cassandra may do great things.

Also, if you put Pulsar into the game, that's another power, because also Pulsar was designed initially with geo-replication. And it was cloud native. So I believe that if you can join Cassandra

with Pulsar, you can do great things, in this direction I mean, of having multiple data centers, multiple clouds.

**[00:20:33] JM:** Do I understand correctly that Pulsar is multi-data center available in a way that Kafka is not?

**[00:20:42] EO:** Yeah, sure, because out of the box it provides geo-replication. That is you can like federate multiple Pulsar clusters, and you can write to one cluster and subscribe to the same topic from another cluster. And then you can seamlessly connect to a remote cluster, and the data is replicated automatically. So you have a replicated remote subscription in another data center. So one of the key feature of Pulsar, geo-replication. And in Kafka, you use MirrorMaker, the stuff that is creating bridges between clusters. And it's not that. That's both.

**[00:21:26] JM:** Let's jog our memories here for people who have already listened to the Jonathan Ellis episode. Why can I make a multi-data center approach with Pulsar than with Kafka? Like why is that technical level?

**[00:21:43] EO:** As I told, in Pulsar, , you can install one Pulsar cluster in one data center, in one region, one cloud, on Amazon, whatever you want. You can install another Pulsar cluster in Azure, wherever you want. And you can connect them. They will share some geo replicated cluster metadata, and they will talk to each other. And then you can connect to one cluster and read and write to the other cluster in a totally seamless way.

In Kafka, you can install Kafka into those two data centers, but you have to set up bridges with MirrorMaker. And then you can connect, create specific topologies in order to see your messages run from one data center to the other data center. And in Pulsar, it is very easy. You just have to connect the clusters and have these global configuration service that usually is implemented with ZooKeeper. And how to say it? It was born with this geo-replication in mind.

**[00:22:55] JM:** Let's go a little bit deeper there. You're describing a fundamental design difference between Pulsar and Kafka, saying Pulsar was designed with multi-datacenter replication in mind. And the result in difference if I want to have geo-replication is, in Kafka, if I

want that, I have to pay the penalty of this MirrorMaker abstraction. So can you tell me what the MirrorMaker abstraction does? And does that cause overhead? Like what does that do?

**[00:23:27] EO:** I must admit that I've never used MirrorMaker in production. So it is out of my experience. I only know this from the docs, because I used to run Kafka but only on a single datacenter instant clusters. Okay? But as far as I know, this MirrorMaker is about explicitly configuring these bridges, these tunnels between clusters. So you have to connect the topics. You have to start one service that copies data from one datacenter to another datacenter. There is no explicit concept of replicated subscription. So you cannot have subscription that is seen from the other side of the cluster. In Pulsar, the data must travel from one datacenter to another datacenter some way. So it's not about the problem of replicating the data. If you want to replicate the data, you have to replicate the data. That's it. But it's more about how to do it, because in Pulsar, there is this shared configuration metadata store that helps you in coordinating all of the clusters. And the pulsar internals, the Pulsar broker deals directly with geo-replication. It’s not something that you add.

**[00:24:54] JM:** So expose us to some of the pain that somebody implementing multi-datacenter replication has to go through. I can't imagine setting this up. So just tell me like what does it take to set up Pulsar that is running multi-datacenter?

**[00:25:08] EO:** Okay. I must tell that I'm more low-level engineer. And I like to work on BookKeeper and discuss this stuff. So I know about geo-replication and how it works. But it's not something that I'm used to talk about. But yes, basically, if you want to set up geo-replication in Pulsar, you have to set up these clusters. And you have to set up a shared configuration store. Typically, you can use a global ZooKeeper installation. But there is a way also in Pulsar to configure your application without this shared ZooKeeper system. And from the command line, you can tell to your Pulsar cluster that it is federated with another Pulsar cluster. And that's it. It's very simple.

**[00:26:03] JM:** So it's not even that hard. What are the failure domains that you hit in multi-datacenter environments rather than single data center environments?

**[00:26:13] EO:** Again, I'm not a very expert of this stuff.

**[00:26:15] JM:** What should I ask you about? What are you an expert on? I mean, you're giving really – For somebody who's not an expert, you're giving very good answers.

**[00:26:21] EO:** Yes, yes, but because, for instance, in this case, I can tell more about BookKeeper, because Pulsar places data on BookKeeper servers. The name of a BookKeeper server is the bookie. And so you can configure the BookKeeper client that in this case is a Pulsar broker to place the data according to a placement policy. And you can use placement policies that are already out of the box that deal with multi-regions. But also, you can deal with fault domains. And this is one of the interesting features of BookKeeper, because you can configure. You can specify many properties about where you want to store your data. For instance, I want to have six copies of my data. And I want that three copies go to one data center or to one fault zone. And the other three copies goes to another data center to another place. And I want to be sure that this configuration is in place. And if something breaks, that is something that fixes the placement of the data and re-replicates data in other places. For instance, if you lose a rack, talking about rack awareness or a region, it's very advanced. And I know about many companies that are running very complex placement policy configurations, like Salesforce, for instance. And Pulsar can leverage all of these features for BookKeeper out of the box, because it is a layer on top of BookKeeper.

**[00:28:06] JM:** So what you're telling me – And this is what I find so interesting about this. So Kafka, I think of as not – It's a super-rich, high-level message bus ecosystem. But it almost sounds like it's sort of like the garbage collected version of what Pulsar is, where Pulsar, you have a little bit more fine-grained control over the resource management, whereas in Kafka you have finer grained control over the API and the application interactions.

**[00:28:36] EO:** First of all, Pulsar is a second generation. So it was born after Kafka. So design Pulsar started from the history of Kafka, okay? But I want to say that Pulsar API is very, very full of features. And you have the Pulsar client API. When in Java, in Python and other bindings, in C++, but also you have a Pulsar IO that is very like to Kafka Connect. For instance, in Pulsar, you have the schema registry that is embedded in the system. So when you are working with Pulsar, you start thinking about, if you want it, you can use it. You can use it out of the box, the

schema registry. So you can use Avro. You can include data with JSON. You don't have to configure serializers, deserializers. Everything is handled automatically by Pulsar.

So the Pulsar API is richer than Kafka API. And coming from Kafka, and when I started to use Pulsar in my previous company, I really liked it. I enjoyed parser, because there were many of the things that I added to code with Kafka. And now like Avro encoding or Avro schema management. And in Pulsar, you have them ready out of the box.

**[00:30:01] JM:** Separate question. I believe there has been a Pulsar company. I know they got acquired, right? The Pulsar company got acquired, didn't they?

**[00:30:08] EO:** There is no Pulsar company.

**[00:30:10] JM:** There is no Pulsar company. Okay.

**[00:30:12] EO:** No. Pulsar was born inside Yahoo.

**[00:30:14] JM:** I realized that, but I thought there was a company that productized it, and I think they were on the show. And it didn't work out for one reason or another. My thinking at the time was basically – Because somebody asked me about this. Somebody was asking about investing in it. And I said I think it's dangerous. And this is like, look, I've built products in the wrong market before several different times. It's been very hard. But my thinking at the time was Kafka has the microphone. Everybody's going use Kafka, or Kinesis maybe, or Google Pub/Sub maybe. But basically, you're going to use Kafka, or Redis Pub/Sub maybe, or RabbitMQ maybe. But basically, you're using Kafka if you're building a platform on top of a pub/sub. And I got what they were saying about Pulsar to the extent that I understood it was different. And I understood that there were tradeoffs and that it had value. And that the BookKeeper abstraction was meaningful. And I was always partial to the Keeper suffix, because that ZooKeeper, the ZooKeeper mascot is probably the best Apache icon I've seen. What's the ZooKeeper mascot? The guy standing there with a shovel? It's really good.

Anyway, but yeah, so I knew Pulsar was good, but it's hard to commercialize one of these open source projects when they don't have all the attention, or when there's a technology that sort of

looks the same, or looks similar, but it's actually not. I mean, what's your thinking on that, that whole Pulsar versus Kafka thing?

**[00:31:36] EO:** Regarding the –

**[00:31:36] JM:** Regarding commercialization.

**[00:31:39] EO:** Okay. Pulsar is still quite young, because it's two years, probably three, that Yahoo donated. And the community really started to adopt it. Thinking about the open source project, it is very good that there are a few companies, bigger companies that started to use Pulsar seriously, like Splunk, like Tencent. Now, recently, we started to work on Pulsar at DataStax. This is good, because for an open source project, it is very good that there are many stakeholders, and the community must be healthy. Also, in Apache, we really take care of all this. We don't want a project that is driven only from one single company. If this happens, the project is going to close or to be rejected by Apache.

From the commercialization point of view, in Kafka, we have Confluent. They're doing a great job with Kafka, with selling Kafka, promoting Kafka. Kafka was new. As I told, I was one of the early adopters of Kafka. But yes, there are other products. And if there is something better, the community will slowly move to something that is better. And if I have to choose my messaging system now, I would like to – And a big enterprise or a small enterprise, I want to use something that as a strong community behind that I'm not locked to work with that vendor. I'm not sure, but I feel that in the Kafka ecosystem, if you want to really use Kafka, you have to use Confluent stuff, like the Confluent schema registry, or many connectors. And they believe that if in Pulsar we are able to keep the community in a very good state and continue to help new contributor. New companies start adopting Pulsar. The pulsar ecosystem would be healthier and it will be easier for companies to adopt Pulsar, because you will not be locked by a single vendor.

**[00:34:07] JM:** What were you doing when the Hadoop revolution happened?

**[00:34:10] EO:** I started using HBase when HBase was in version 0.9, probably. I was working in my email marketing company, and we needed to scale out. And I started watching at Hadoop. And then I adopted Hadoop. But not Cloudera, because we didn't want to pay a vendor,

because it was all open source. And in my previous company, and now in DataStax, I always like the open source because I can – Especially if there is a real community beyond the project, because you can ask for problems. And, okay, if you have some commercial support, and you know that it is up there. And if I use HBase, or if I use Cassandra open source, I know that there is Cloudera, there is DataStax, there is some vendor that will help me in case of I have a very big problem. But most of the cases, those projects run well out of the box.

**[00:35:26] JM:** So like when do you go all-in on open source and just say like, "I'm not working email marketing companies anymore. I'm going to go all-in and open source software." Was that a hard decision? Because I mean, the open source software that runs our world is built by really wide range of people. And the people who are all-in on an open source project, a lot of times they're working on the open source project outside of their work, and then the company that's basically sponsoring the project contacts them and says, "Hey, do you want to do exactly what you're doing, but get paid for it?" And then they just get hired and they get to work on their open source project of their dreams. It kind of looks like that's what happened to you.

**[00:35:59] EO:** Oh, yes. This is what happened to me. Because I started to work in my previous company at **[inaudible 00:36:06]**, we started to use a lot of open source of software, for instance, ZooKeeper, BookKeeper, and Kafka, and HBase. And everything that starts with Apache, it is in Java, probably. Yeah, I started to contribute to Tomcat. My first batch was for Tomcat, for instance. And I entered this world. And also, I've been driving a team of engineers. And I always told them to contribute with opening an issue or contributing a pull request. And then you work – You get in touch with many engineers from all over the world, people that are better than you. And then get in touch that you learn many things.

I come from a small company. And working in Apache, it was a great step for me, because I got in touch with really great people, great engineers. I learned lots of lots of things. And also, I started a few open source projects in my company, because I thought that open sourcing things that are useful for other people, and they're not in your core business. For instance, we open sourced a database, a distributed database in my previous company **[inaudible 00:37:30]**. It was an opportunity to meet other people, connect with other engineers, and also to get feedback from the community and probably people find blogs before you. And then, yes, I started to receive a few offerings for a job. And then one day I decided to start working everyday

on open source, and now in DataStax. So that was my story. And I'm very happy that happened. And I'm happy that now I can contribute to the Pulsar community and BookKeeper community with my work every day. And part of my work is to be part of the community and help the community to be in very healthy state.

**[00:38:20] JM:** Community, okay. What's going on in the Pulsar community? Community management is tough. The whole community management on Kubernetes has always been interesting to me, because it's so big. It's so shared and so corporate. Tell me about community management dynamics in open source. Give me your lessons.

**[00:38:40] EO:** You have to be inclusive. You have to help anyone that get in touch with the project to be able to speak. Because many people – Really, I know, sometimes you have a problem, and you are shy. You don't want to send a message to the mailing list, “Hey, I have a problem,” because you you're afraid to look stupid or to contribute a patch.

So a big effort from a community management perspective is to be present on Slack, on the mailing list, to give answers to let people feel that they are welcome to the community. And because I know that if an open source project doesn't have users, it dies. And I always say that if a project doesn't have bugs, it is a project that no one uses. So if you started open source project and your GitHub issues tab reports zero issues, you're sure that no one is using your software. So it's not really open source. It's not a community. And so it is very important to help the new people that try to interact with the project to feel welcome. And then they can grow up. They can become committers. They can take responsibility on the project, because being a committer on a project is not always being able to write code, to be a super great engineer that knows about distributed systems, or disks, or this stuff. It's about caring about the project.

For instance, I left my previous job. And now I'm continuing to support the open source project that I was part of. Not only the project that that now I am working on Pulsar and BookKeeper, but also I'm part of other open source projects. And this is very important. The community – One of the motto of the Apache Software Foundation is community over code. And I'm very happy that more and more, we are accepting new committers that are writing code documentation. They are preparing events, community events, because this is very important. Open source project, at least in the Apache Software Foundation, they are more about the community.

**[00:41:18] JM:** All right. Well, we've touched on a lot of areas. Can you take me inside the Pulsar right now? Take me inside what's the hardest engineering problem that Pulsar is working on right now, that Pulsar community is working on?

**[00:41:31] EO:** We recently released a big feature. It is transaction support. It was a missing feature in Pulsar. We're working hard on this. But this is the most interesting feature that we're working on. We are also working in reducing a little bit of tech depth. Because even if the project is very young, the project grew so quickly during the past two years that many, many, many features were added. And so when you have a very large codebase, you have to take care about many things. For instance, we spent the past six months reducing problems with flaky tests, for instance, because if we have many flaky tests and you run for request validation, then the pull request validation phase, and new contributors that come, you send a patch, and CI phase. And you don't know why and you get frustrated. And also if you have many flaky tests, you're not confident with the release. So that wasn't very interesting problem. A little bit annoying. But it's very interesting, because we spend much of time in trying to find the good way to fix this problem.

And another interesting engineering work that is happening is about abstracting from the ZooKeeper API, because Pulsar runs with BookKeeper and ZooKeeper. But there are many places in the code that are referring directly to the ZooKeeper API. So there were many problems. So we're introducing an abstraction layer over ZooKeeper, that it's not about removing ZooKeeper. It's about reducing tech debt and having a better way to see the code of Pulsar. And yes, it's an opportunity to get rid of ZooKeeper if you want to get rid of ZooKeeper. We already did this in BookKeeper project.

And refactoring code, it's very hard, because in this kind of project that are very mission critical. Even if tests are passing, you don't know, because sometimes you inverted two lines of code. And when you run that code at scale, you can see disasters. I saw that things in production, one line of code that every test was passing, everything look good, code reviews, but when you run in production at scale, it crashes. So making code refactoring is good. Reducing tech debt is good. And we must read. But it's dangerous. We are doing our best to make Pulsar rock solid, stable and more stable. In this way, everybody can count on pulsar.

**[00:44:35] JM:** Do you have a set of acceptance tests? Or can you tell me about the integration testing system and how you make it decentralized? Because it has to be decentralized across an open source community?

**[00:44:45] EO:** Sure. We have at least three layers, the open source project has its own CI. And before accepting any pull request, we have to see all tests passing. Tests are about unit tests, integration tests. So you start the service. System test, you start the service with Docker and things. This is usually not enough for a system like Pulsar, because you cannot simulate a cluster of 5, 10,100 machines only on one CI machine. So other companies, for instance, in my team, we are running additional tests. So we are running the same test in CI. But I gave a talk last week at Pulsar Summit about the distributed system test framework that we have at DataStax. And I contributed to it in order to use it for Pulsar. The name of the system is Fallout. This is basically about running real distributed tests, that is about you start a real cluster of machines on Google Kubernetes engine, or in OpenStack, or whatever you want. Then you install a real cluster of Pulsar with 3, 5, 10, 20 machines. And then you run a workload and you verify automatically that everything went well. You can inject failures. So these kinds of tests were missing in the Pulsar open source project. And now, in my team, we are working to contribute this test to the community. But we are already running this kind of test at DataStax, because we want to be sure that Pulsar really runs well and you can count on it. And it is not enough to see the test passing, the unit test passing or integration test pass. You really must have a way to see every day to launch clusters with machines and run Pulsar and simulate real workloads. And you can see that every commit in the Apache Pulsar master does not break things. And it happened three or four weeks ago. There was a regression. And we saw it because our tests started to fail. Test that send like 1 million messages with 100 partitioner topic, and they started to fail. And so with a community, we found the root cause. And we asked the community to find the fix. Actually, I did not find the fix. Other engineers from other companies found the fix that contributed the fix. But you know, it's a community.

**[00:47:32] JM:** This has been an illuminating conversation on a lot of levels. So Enrico, thank you for coming on the show. Is there anything you would like to close out the conversation with? I know we just touched on a lot of different topics. Is there any point you'd like to underline that you were really hoping to come across in this episode?

**[00:47:47] EO:** The best thing that I can say that I really enjoyed that Pulsar is a healthy community, and the community is growing. And I hope that everybody in the ecosystem will do its best to make Pulsar a great project with many users and to be something you can trust on and you can build real enterprise level systems.

**[00:48:17] JM:** Awesome. Enrico, thanks for coming on.

**[00:48:20] EO:** Thank you very much.

[END]