EPISODE 1297

[INTRODUCTION]

[00:00:01] JM: Julien Le Dem has been in the field of data engineering for quite a long time. What is the background of Julien Le Dem? Julien Le Dem helped start the Parquet Project, as well as the Pig Project, the Arrow Project. He's worked on Heron. He's worked on Kudu. He's worked on Tez. So Julien Le Dem is one of these guys, like Flavio Junqueira, or perhaps like Ben Hindman, or Matei Zaharia. Basically one of these people who has contributed a momentous amount of technology to the open source data world and just has essentially a universal respect. There are several of these people who have been around data engineering, let's say since the Cloudera days, like early Cloudera days, and they've done so much work that the entire industry just reveres them and views them as artists. So Julien Le Dem is one of those people. Julien Le Dem is a fantastic guest.

What is Julien Le Dem doing today? He's working on Datakin. So what is Datakin? Datakin is data lineage. If you've listened to any of the shows we've done with Pachyderm, for example, Pachyderm is a data lineage company. There's a few other data lineage related solutions. So data lineage is something that's really, really hard to solve. What is data lineage? Data lineage is like what's the history of your data set? So like why is data lineage important? Let's say you're building a product that issues loans. If you're issuing loans, and you reject somebody for a loan, if that rejection turns out to look discriminatory, then you may get investigated for discriminatory lending practices. If that happens, you want to be able to defend yourself. If you want to defend yourself, you need data lineage, in many cases.

So imagine being interrogated by somebody, by some sort of loan regulation officer. And that loan regulation officer says to you, “Okay, look, Jared got rejected for the loan that he applied for from your machine learning loan giver system.” What gives? What why is that what happened? And you say, “Well, it's actually just the data. Like it's not us. We’re just training our algorithms. And you can take a look at our training data. We just have bias training data.” You want to be able to prove that. And that's going to be like a big problem in the future. So data lineage is basically this foundational level engineering problem.

And so why is data lineage hard? Can't you just store all your data? Like can't you to store or revision history of all your data? No, of course not. That's insane. Let's say you've got a terabyte of data, and that data changes over time. Let's say you have a terabyte data set of user profiles. And over time those user profiles get updated, right? You've got a bunch of updates that are coming in, CRUD operations, that are changing the user profile data set. It includes like kind of clothes you wear, and what kind of hat do you wear, and colors your skin? And do you have freckles? And do you like Aerosmith? So you've got like this big set of user profiles just sitting in a data set somewhere. And that data is changing over time. You're updating your profile. Let's say that's the only thing your application does. Or let's say these are user profiles of people who may or may not want loans in the future. So you've got a big set of attributes, including skin color, and including age, for example, including income. So you have all this data. And basically your data is your user profiles, and you're going to try to decide if you need to lend these people money.

So you've got this data set, you've got this system that you're just trying to figure out how to lend people money. And then one of these people asks for a loan. And you've got these machine learning models that are being trained on this data set repeatedly. Oh, maybe they also have like their payback rate for loans over time. Let's say like this is a high-volume lending platform. So like these are people who borrow money on a regular basis. So you have a loan history. So over time, you have a loan history. You have a profile. You have the changes of the profile. These are the things that you would theoretically need to save, because like you're going to continually want to run machine learning jobs not just on the state of the human today, but like how the human has existed in the past. So that's how foundational of a problem this is, because there's a big, big, big question around how can you compress across time? What is compression? Compression is the reduction in size of data through common attributes of that data. If you have repeating attributes in a data set, you can compress it.

So think about it. If you've got a profile over time, probably there's a lot of repetition there. Probably if it's keeping track of what color hat I wear every day. And let's say there's like 300 – Over a year-long period, let's say there're 300 instances where I wear a white hat, and 65 instances where I wear a red hat, two kinds of hats. Just if you look at this is a time series, there're only two kinds of hats. But the word like white hat might get repeated 300 times. The word red hat might be repeated 65 times. You should be able to compress that, right? So you

should be able to compress that information across time. That's important, because otherwise your data set just blows up. So that is what Julien Le Dem demo is working on.

We barely talked about that, if I recall the conversation, but like it just gives you an idea of like what kinds of problems does Julien Le Dem have a taste for? It's those kinds of problems.

[INTERVIEW]

[00:05:35] JM: Julien, welcome back to the show.

[00:05:36] JLD: Thank you. I'm happy to be here.

[00:05:39] JM: Yeah, so we've done one or two shows. I can't remember, maybe three over the years. And I think of you as somebody who is a data engineering, almost historian. The field of data engineering is pretty young. I think of it – At least I'm sure other people will critique me for this. But I think of the data engineering revolution is basically starting with Hadoop and HDFS, and then Cloudera, and all of the downstream impact of that all the way to today, where you have much better data pipelines that still have tremendous problems. So why don't you give me your condensed history of the evolution of data engineering up to today?

[00:06:20] JLD: Wow, condensed.

[00:06:21] JM: Condensed into like two minutes or less.

[00:06:26] JLD: Yeah. So I guess I've been in data for more than 10 years now. So I guess maybe that apply to the timeframe of everything. So when I started, so I was at Yahoo like more than 10 years ago. And so that was the place where the whole Hadoop ecosystem kind of started, where it started growing. And so it evolved quite a bit, right? Like back in the day, you would have HDFS, so sharing files, and then using MapReduce jobs to transform things. So pretty simple tools. And I think since then, it evolved quite a bit. So I think we remember there was an article by Stonebreaker, like the famous database researcher, about like why going back to MapReduce was doing everything wrong. And it was a huge step backwards. And in some way it was, but at the same time, it was the foundation that enable like, nowadays, there's a lot

of the database principles that have been re-implemented on top of those foundations. And the Hadoop ecosystem has grown quite a bit. And even to the point that there's less and less Hadoop in it anymore. But all the project that grow around it is still very much active and a lot of things have evolved.

So I think, recently, what you're seeing is SQL is a lot more part of the picture as well, whether it's the SQL and Hadoop, or Spark SQL, or the SQL and streaming that's happening, or whether more cloud native warehouses with BigQuery and snowflake and those kinds of things. So it's become – Like there's like been a huge growth. So Hadoop was very JVM oriented. And now there're lots of Python, there's lots of SQL. They're looking at combining machine learning oriented API's. And also having a lot of analytics people, or doing transformation is SQL and building dashboards for analytics. So there's lots of fragmentation of many different tools.

And so one of the things we're seeing is how do we make everything work together, right? Like there's the DBT is a new way to build dependencies and transformation for the SQL world. And you still have things like Airflow, or Spark that are more like programmatic API's. And so we have to figure out how to make all these work together. It's kind of big expansion of a lot of different tooling and specialization in the field, right? Like data engineers used to be people who would write MapReduce jobs, and maybe be able to contribute to Hadoop for those projects, and build a platform, and build the transformation. And nowadays, there's a lot more specialization. Like enabling people will do SQL, a lot more math, more analytics. People who do more machine learning, people who do more platform. And so this kind of very different space than it used to be like 15 years ago, I guess.

[00:09:28] JM: I have a counterfactual history exercise. Imagine a world in which Yahoo would have poured significant resources into in-house Hadoop-based technology that they could sell to the market. Do you think Yahoo could have been saved by focusing a ton of effort on the Hadoop ecosystem instead of kind of doing this externalization thing where they did Hortonworks?

[00:09:54] JLD: Yeah, I don't know. In some way it was a bit they did Hortonworks, but Cloudera existed already at the time, right? So there were things like – It's a question of DNA. I don't think it was in – Like Google, for example. At the time, there was this joke that would go

around like, “Oh, what do two Yahoo engineers talk about when they meet?” right? Well, obviously, they talk about Google, right? That was big for Facebook. Started like eating house lunch.

And it's kind of be tough company DNA. Like Yahoo was very much – And I'm sure, people will have different opinions. But at the time, there was a lot of different – Each Yahoo product was almost its own independent things. And even each Yahoo in each market was its own little organization, which makes it like pretty agile in moving in each market, but not very much having a strong culture of having platforms. And like Google, really, like more like overarching platforms internally. And so there were things like Hadoop came out of the search organization of Yahoo. It started being used in other areas. And so you could argue that Yahoo could have been the service-oriented platform, but didn't work that way, right? Really, I mean, they could have – I don't know if that was really in the DNA of Yahoo. They were very much focused on how we make products for our users, how we build that in an agile way. And in some ways, the things like – It's kind of social media, I guess, is the term that Yahoo didn't take from the consumer perspective. And in that sense, like many Facebook to cover, and even Google wasn't able to compete in that area. But Google won the search area. So I guess they're fine.

I don't know. It's hard to think. I think like it was not in Yahoo's DNA to build that. And I think if they had done spin up, spun-off Hortonworks engineers, there was a lot of demand for Hadoop committers at the time, and Cloudera existed. So there was a risk that people would have left, would have done all the things.

[00:12:11] JM: So you got into data engineering before it was even called data engineering. You were the coauthor of Parquet, which is a file format for Hadoop. So you've been deeply involved in the ecosystem. And you used some of that domain expertise to work as principal architect of Dremio for a few years. Dremio has always intrigued me. I feel like Dremio got to the data engineering world at a very unique time in place. How would you describe the product vision for Dremio?

[00:12:44] JLD: Yeah. I mean, that's a better question for the cofounders of Dremio. The way I got into Dremio, at Yahoo, I build platforms on top of Hadoop, and then started contributing to the open source projects in that area, in particular, Pig, which led me to Twitter. And at Twitter,

that's where there was the Hadoop and the vertical space. And that's how I got into starting the Parquet Project and getting Hadoop closer to the database. And like Vertica was lower latency, but couldn't handle as much data as Hadoop could. And Hadoop was very high latency querying. And so we were trying to bridge the gap a little bit. And that's what led me into joining Dremio, right? Like I think the Dremio vision were very much aligned in how do we make the whole data ecosystem more like a database, right? So you can use Dremio and you can query everything as if it's one federated database. And the Dremio magic lets you define views that will enable quickly querying things without having to change your model, right? So you stick Dremio in front of your data lake, or all the things you may have, whether it's MongoDB, or Elasticsearch, or other kind of services. And it looks like one big databases. And there are things you can do in trivial so that it becomes performant so that you can use Tableau, or your BI tool, Darkly, on top of it, right? So it removes a lot of the complexity of like the typical model people have and still have, is you have the data lake, which is still very high latency. You can't really curate in an interactive way. And you push a subset of the data or some transformation of the data into more traditional warehouse, where you can stick your BI tool like Tableau or Looker on top off and get visibility. So it's kind of that part of that vision. And really, I'm sure that vision evolved over the year, is to really abstract out of all that complexity and make it like one database. And of course, depending on where the data is stored, the speed of response is different. But you can do things in Dremio that make things fast for your specific use case without having to have a lot of different transformations to maintain.

[00:15:08] JM: There was this guy I interviewed from Microsoft a while ago, and he had an idea, or maybe he didn't have this idea. But he talked about this idea, the term virtual data. Have you heard that term?

[00:15:20] JLD: It sounds like it invocates several things to me. But yeah, like keep going.

[00:15:25] JM: Yeah. Well, I mean, so his idea that he really kept emphasizing as virtual data is essentially the idea that you build some kind of indexing around all the data in your entire data world. And then you have various access patterns that allow you to access that data more efficiently. And I always thought of a Dremio as kind of a virtual data company where you're supposed to basically use – Like Dremio offers you this middleware system that makes it easier to access like heterogenous data from lots of different sources.

[00:15:59] JLD: Yeah, that's right. So the first aspect is like federated, right? You can query the data where it is into one SQL engine, and you could join your MongoDB table with your S3 dataset, if you want. But of course, this is limited by the performance of the underlying storage. And so Dremio virtualizes the soul like indexing. You could call it indexing, or projections, or different type of rewriting the data internally to make it easier to query. It's a little abstract at, like it's all virtualized. Like people don't need to know of what transformation happened under the hood to make things more efficient. More like lower latency queries.

[00:16:42] JM: Okay. So back in 2015, 2016, 2017, I was thinking that the future of data engineering was we've got these data lakes, we've got these streaming systems, we've got Kafka. And all the action is happening in the streaming systems, or streaming all the production data through Kafka. We're reacting to it. We're doing Hadoop jobs, and Spark jobs, and Apache Heron jobs, Apache Storm things on top of all this moving data. I thought this was the future. And then it turns out that the future is actually you put everything in a data warehouse and just do whatever you want with it. Why did that happen? Or was that just a detour? Are we on our way to like better data lake support and more interactive systems?

[00:17:29] JLD: I think the proportion of streaming is still increasing. Like people use streaming more and more, and it used to be a few percent of their jobs would be streaming. And it's growing to – I forgot. Someone was giving me number recently, like 10%, 20% of the jobs of the logic in a data analytics is training. I think both are useful. I don't think we're going to replace batch processing entirely with streaming. They're different, and they're complimentary. And probably what will happen over time is like those different engines will be less different, right? You can see a lot of unification in the batch versus streaming world, like Flink, Beam, Spark. The old look at being streaming and batch platforms. And in a lot of ways, depending how your streaming job is defined, you can rerun it in batch as well, and make it run faster if you want to run it on historical data.

I think some of the aspects, those things, they're different beasts in some way, right? The way you use streaming is you want to have low latency dashboard being updated. But you still need a warehouse and being able to do an analysis. And because you may want – As a data scientist, you might want to look at the past six months of data, and you might want to look at

the data, and “Oh, we need to figure out what are the outliers.” And then once we figured out what the outliers are, we may want to remove them from our analysis, because there may be bots to remove, or they may be different type of users. And we want to have an analysis that's more precise. And we want to understand some of those things. So in that kind of case, this kind of an exploration of the data, right? You do a query. You look at the result. You transform your query. You may be selecting a subset of your data for doing more analysis. And so it's very different from a streaming system where, okay, you know exactly what you wants to compute all the time in a very low latency. And you may have defined KPIs for your business, or you may want to use the data to improve features in your product. Like typically a recommendation engines are a common data-driven feature. Like we use the data we collect to improve the product.

So they're really fundamentally, I think, different things. And I know some people kind of advocate for everything should move to streaming, and so on. But I think you use them in different ways. It makes a lot of sense for me that you would want to use both and in different ways. And it makes a lot of sense to me as well that streaming system showing more and more capable of doing both streaming and batch, and kind of letting you either rerunning, but in a way that abstracts it to the user, right? If you want to change the logic of your streaming job and restart it in the past, you should not have to rewind the entire stream and read it in order as if it had happened. Like depending on the logic of the job, you can actually have a different plan executed to compute the historical data and make it happen in parallel, right? Depending on typical aggregation windowing algorithm, don't need to be run in order. You can run in parallel for different periods of times in the past, which is exactly how batch processing chat works.

[00:21:07] JM: Zooming out to the application level. You worked at WeWork for a little more than two years, from 2017 to 2020. And WeWork is a company that has basically a bottomless amount of data that it could collect. If you think about 2017 through 2020, WeWork, how popular they were, especially pre pandemic, how much action there was going on in it, all the telemetry data that could be collected. So walking into WeWork, I assume their data engineering was not very well developed in 2017. What did you do over those two years? What was your strategy to grapple with the scale of that data?

[00:21:49] JLD: Yeah, so that's interesting. So before WeWork, I was very much like at Twitter and Dremio later. I was very much drilling into query engines, and like going deeper in this area. So it's kind of more vertical growth in that space. And then when I joined WeWork, I joined as the architect for the data platform. So it's kind of more like broader scope. And at the time, WeWork was relatively – Either a relatively small data engineering team. It was very much centered around the data warehouse. So doing SQL queries, SQL transformation, ingesting the production data into a warehouse and building around it. And there were a lot of ambition at the time about how we understand how people are using the space, right?

So when you use a website, of course, when you use Facebook, whatever you click on, Facebook knows everything you click on, everything you look at, because everything goes through your computer. Where you're in a physical space, it's a little more different, right? So as part of WeWork, and I don't want to misrepresent some of – So it's kind of take it with a grain of salt. I was just one participant in all of that. But as part of WeWork, there was **[inaudible 00:23:06]** of how do we algorithmically improve the layout of the buildings, right? It's kind of WeWork was doubling every year. It's like, at the end of the year, they add more, twice as many building, if not more than they had to start with. And so because there was a heavy emphasis on how do we algorithmically improve the design of the space, right? It's kind of – And they published a few papers about layout algorithms and how we improve this. How many two people office? How many five people offices? How many 10 people offices? So how do we lay them out?

And so the goal was also how do we collect data? How does spaces use, while also respecting privacy of our users. Obviously, on social media, one of the problems is Facebook, or other social media, they know everything you're doing there, and they know who you talk to, what you talk about, and everything. And so in a physical space, you want to be able to understand how people are using the space was preserving their privacy. And so you want to collect data about how meeting rooms are used? How common spaces is used and so on.

And so during these hyper growth phase, it was really about how do we enable the data collection, the processing, all the things people want to be able to do? And it was not only using a data warehouse and building metrics or analyses, but also enabling all those things, right? So it typically involve like stream processing. How do we ingest data? Especially IoT data, like you

have IoT devices that collect information about this space in various physical location. How you ingest that streaming data? How you archive it? There's the data emotion versus data trust notion. How you consume to do streaming transformation folder. Retime aspect you were talking about. How we do batch processing? And in the batch processing, you have all the areas of more analysis-oriented, ETL or the transformation for building a model for enabling analysis matrix and understanding the space. And then there's all the machine learning oriented processing that feeds into how do we improve the layout algorithms by learning how people are using this space? How do we improve pricing for the rent of an office space, and so on? And so building all those things.

So for those two years, we spent time enabling all those other use cases, not just SQL and the warehouse, but the collection, stream processing, batch processing, BI on top of that, and making sure everything works together. So that was a process of what's the best. Like, obviously, let's not build everything in-house. So it's kind of picking the best open source project or the best project for everything, which led us to pick Kafka for ingestion, Airflow for scheduling, looking at Spark and Flink for batch streaming processing, Snowflake for the warehouse, and Apache Iceberg for the storage at try still ready. And making sure we have schemas. We have good contracts. We enable – In a modern company, data-driven company, like every team in the organization uses data in one way or another. So you have a lot of dependencies between team. And so that led us to the one piece that was missing, right? So kind of taking the best of breed open source project in that space. And in our opinion, one piece that was missing is this open source environment. And that led us to creating Marquez as an open source project, was the understanding the data management, understanding of metadata. Having somewhere where you understand all the datasets that exist, all the job that exists that transform like read data set, produce data set, all those transformation. And understand the lineage graph on how they all depend on each other.

So that was really the start process of creating markets, right? Because I don't think the Hadoop ecosystem had already produced something that would allow that. And that was really critical. When you have many teams that consume and produce data, and they depend on each other, everything becomes brittle. And like people are worried about changing anything, because every time they change something, they break something. And they may not even know that the thing is broken before a week, or someone complains that something's broken. And because people

have very detailed visibility, like a few hops removed from what they're doing, right? Like they may be reading a data set, but they don't know who's producing it. Or they may be producing data, but they don't know who's consuming it. Or they may know the first level, but they don't know how many things downstream from them are depending on it. So this kind of the sub-process.

So for two years, yeah, we build the all the data platform, enabling all those use cases, like streaming, batch processing, machine learning, and enabling this organization that was doubling every year. And it's a really fast pace growth. And it didn't end the way we wish it would. But WeWork is still around, and it's still kicking. So I don't think this story is – We heard the end of that story yet.

[00:28:27] JM: I actually agree with you. So I saw the WeWork documentary. And I think a lot of people walked away from that with a negative view of the company. I actually walked away with a very positive view of the company. And I actually see a lot – I still see a lot of potential for the company like given how much work is changing, I don't think we really know where things are going. I don't envy the teams responsible for having to unwind some of the real estate commitments that they have. But obviously, the core brand is still really strong. I like the founder. I thought he was like really charismatic. Like if you want to be following somebody who's like basically trying to take over the world of commercial real estate, you want this guy that looks like Jesus and sort of sounds like Jesus. Like that's the kind of person you want in charge, but quite a personality. Any thoughts on leadership from – Or you don't even have to talk about that if you don't want to? I know we're talking about data, but I'm just curious. As somebody who had a firsthand view?

[00:29:26] JLD: So I haven't watched the documentary yet. I'll definitely watch it. So I think the WeWork product is great. And I think, from a business perspective, they weren't building – Doubling the number of buildings every year. And they were filling them, right? The reason they were doing that is there was demand from it. So, of course, the tactic is very high risk, right? Because the investment is so important. And when you start a new building, you start pouring money in the construction, like the interior design and like implementing it. But you don't make any money on it until you open it. So, of course, WeWork was losing vast amounts of money every year, because they add more buildings and constructions than they had building and

making money. And I think, in the end – I'm not going to comment on the leadership. But in the end, the thing blew up, because this strategy was working as long as money was being poured in the company, right?

And at some point, if there was not the same amount of funding coming in, we would need to slow down that growth to allow for the company to be self-sufficient. I think it's kind of a risky thing. It's like being always on the edge where there's a high risk of stumbling, which is what happened, right? Because it's kind of what you look at. But looking back, you would think, "Oh, in that situation, with all those buildings that were in constructions," at the time, WeWork stopped starting new projects, basically with a failed IPO. But they still have a lot of momentum, lots of inertia in all the projects that were ongoing. And the pandemic hit not that long after that.

And one thing to recognize, and they're still around. So I think the business is kind of sound, the product is awesome. Yeah, of course, I think leadership – There's a value in always running forward really fast. The problem is if there are unexpected obstacles in the way.

[00:31:42] JM: Okay, look, we're going to get to Datakin, because I want to talk about that. But just as a little bit more on this point, because you said about this like, "Okay, you can't do the WeWork thing," at least in the way that they were doing it. But there is some science around capacity planning. Like, clearly, we're able to build out data centers in a sustainable way for like somehow. I have no idea how they do it. But somehow there's always enough capacity at AWS. I don't understand where were the people who are doing like capacity planning in a sustainable way. Like how did – I guess if you just have leadership that's ahead of their skis, literally, everybody else in the company is going to get ahead of their skis to.

[00:32:19] JLD: Oh, that's what leadership is for, right? Yeah, I mean, if the message is don't worry, keep writing. It's just like – Yeah, I don't know. But obviously, in the end they've kind of like, "Oh, yeah, we're going to IPO. And this is where the funding is coming. And this didn't work." So yeah, it's just a high risk situation that didn't work out. So, I mean, in some way, it leads to Datakin, right? Like because –

[00:32:50] JM: Okay, great. Take us there.

[00:32:51] JLD: because the fact these failed IPO and the fact that – Yeah. So definitely the priority of the company was not on high growth anymore, and a lot of division had to be scaled down. And maybe there was not so much need for a lot of the stuff we built over those two years. It's kind of created the situation where like, “Okay, like, what do we do next?” right. So we started Marquez, because we very much wanted to be not building propriety stuff, but be part of these data ecosystem. Like used to be Hadoop ecosystem, but I think people use Hadoop less nowadays. But there was a missing piece. So really like starting an open source project and starting Marquez was our way to say, “Okay, this is the missing block.” And the only way we know of like improving the ecosystem is contributing to it, right? It’s kind of you have two ways, right? You can build things in house on top of open source, or you can start contributing project or contributing to existing open source projects. And the advantage of contributing those things as open source is they become part of the ecosystem.

And if you look at Parquet, for example, or Apache Arrow that we started at Dremio, they really became part of the ecosystem to a point. Like, nowadays, Parquet is pretty much everywhere. You can read/write parquet from BigQuery. You can do that from Ssnowflake. I think BigQuery returns results set in an Arrow format, if you want. And all of those things would not have been possible if Parquet had been a Twitter proprietary fund format, right? We would have had to integrate everything. And so really making it open source, collaborating with Cloudera on it, was really a goal of, “Oh, this should be part of the ecosystem.” And if you manage to bootstrap an open source project like this, it really takes a life of its own. And that's the value of not just making things open source, but adding them, like donating them to a foundation, right? It's really about saying, “Look, this open source project, it is owned by the community. It's driven by the community. It's not owned by an entity in particular. It's not owned by an individual. It's really something we own collectively. And the more people contribute to it, the more valuable it is, right? Or more people integrate with it, the more valuable it is. So Parquet became more valuable the more project started to be able to read and write to it. And similarly, Arrow follows the same dynamic. And Marquez and Open Lineage follows that same logic, right?

So we started with Marquez at WeWork and making it with open source because we wanted to enable the same ecosystem, right? Marquez is the kind of project that will become more valuable the more people integrate with it, the more it's used, the more it's adopted. And that's also what led to the birth of Open Lineage, because Marquez has two parts. One part is the

ingestion of lineage. How do we collect the lineage? And the second part is how do we store it, and analyze it, and be able to navigate it? And so that's where we started Open Lineage, because the conclusion was, “Well, the standardization of lineage, and just the collection, being able to emit and collect lineage is the most widely applicable part of that project.”

So speaking of Open Lineage is a way to say, “Look, this is the most widely applicable piece.” This is really something that will benefit the ecosystem, because there's one big gap of understanding lineage and how we collect it. And there're many different use cases for using lineage. People care about governance, compliance. From a compliance perspective, like, “Oh, where is my user private data going?” Like I need lineage to understand that. There's like lots of banking regulations where like you need to be able to prove that the result of the transactions you're computing are this error derived entirely from the data you're collecting about what the transaction are doing. So really collecting lineage and proving that the data flows where it should is really important.

There are operations and data pipelines observability. All those aspects of how we do data ops properly. How do we operate our data pipelines properly? Also records lineage to understand how things depend on each other. And so there are all those use cases that may spawn different tools for solving them. But they all need collection of lineage. And so we started opening it the same way we studied Apache Arrow back when I was at Dremio. It’s like, “Look, we're considering there's this missing piece, right?” We ought to have a standard way to collect lineage and for everything to expose their lineage, whether it's a data warehouse, like Snowflake, or BigQuery, or whether it's Spark, or whether it's Flink, or it’s a streaming system, or a batch system. They should all expose their lineage in a standard way. And so we reach out to a bunch of people in that community, right? Like from all that work in the Parquet community, in the Arrow community and others, you keep meeting the same people, the people who care about open source. Like people who care about building these great ecosystem. And 90% of the people we talked to were like, “Yes, we need this standardization of lineage. Why don't we have that already? How do we make this happen?” right?

And so that's how you start a project like that. All you need is kind of planting a seed. We need to get the project started so it happens. The reason we don't have it already is because we didn't put the effort of getting together and starting building that, right? So that's kind of how we

started Open Lineage last year, spinning it off out of Marquez. And so Open Lineage and the name is really meant to draw the parallel with open telemetry. Open telemetry is how you collect traces, and metrics in the service world. Open telemetry is part of the CNCF. And Open Lineage is part of the LFI and data, which is the other Linux sub-foundation. You have the Linux Foundation. Under it, you have the CNCF for the cloud Kubernetes stuff. And you also have the LFA and data, which is the other Linux Foundation, sub-foundation, for all things machine learning and data. And so really, the Open Lineage is the open telemetry for data pipelines. It enables collecting that lineage in a standard way, understanding this ecosystem and creating this observability of everything that happens in your data platform.

[00:39:58] JM: Alright. So, lineage, this is something that people have wanted for a pretty long time. And the use case I always think about with lineage is let's say you're building a big data system that calculates whether or not somebody can get a consumer loan. If you want to offer people loans based on data science, you want to be able to have an understandable history of how you arrived at that decision. Lineage gives you the language and the data to express what calculations lead to a certain decision. Am I giving a reasonable description of lineage, or a reasonable example of lineage?

[00:40:49] JLD: Yes. So if we look at it – I mean , and this applies, you're talking about giving loans, and if you're doing a research on COVID, and the efficiency of which vaccines works the best on how this is spreading, or things like that. This is also similar use cases. People can use data to optimize ads, or they can use data to save lives as well. I think we tend to give examples of like the very commercial aspects. But kind of there are lots of different ways we use it. And so if we look at it in a kind of meta way, those things become complex. We talked about big data. And I think for a long time, people focused on big data. How do we make things scalable? How do we process more and more data efficiently? But, actually, another big problem of big data is not necessarily just the sheer volume of data. It's how many different datasets we have. How many different transformations? How many different people are involved in transforming that data?

And so if you look at lineage, it's not just someone getting data in and doing an analysis and producing a report, right? The fact is, is there are many sources of data that are being used. And then there's some kind of first levels of transformation that adapted to kind of making it

better to analyze. And maybe another level is kind of combining datasets with another one, right? You may be merging from different data sources. And then there would be other levels of transformation that do anomaly detection, removing outliers, removing bad data. There's kind of a lot of complexity and a lot of layers, many layers of transformation, before someone actually starts using the data for doing analysis. And maybe they're making a decision on approving your loan. Or maybe they're making an analysis to understand how a virus is spreading. But there's lots of complexity. And all those things are constantly changing, right?

Like when you use a web product, and the new features are rolled out, or new things are happening, the way data is collected is always constantly changing, although things evolve all the time, which makes all those dependencies and all those complex layers potentially prone to breakage, right? Like someone change something. It broke something down the line. And if you don't have a clear understanding of that lineage, it creates problems.

So from an operations perspective, you may have your data is actually hard to trust, because it might be broken. It might be wrong. Like some logic has changed and you don't know either resulting data is still correct. Or you might be delayed, and you have to do your analysis on stale data. And from other perspective, if you go back to the privacy use cases, nowadays, we have better regulation with GDPR and CCPA of, well, companies have to know where their user data is going. And they used to not do, right? Like when through GDPR first came out, like companies were like, “Oh, we need to know where our user data we're collecting is going in all our systems.” And they realize that they don't, right? Like because the way this works is very organic. People need data for analysis. They start reading from somewhere. And if you don't draw a map of all the dependencies of what reads what for what purpose, then you don't necessarily know at a macro level where the user private data is going. And so you cannot guarantee that you're not using the data for purposes that was not intended by the user to start with, right? Which ends with having much better system like when you use a social media. Now you can select to opt out of certain advertising. You can opt out to a bunch of things. And that's made possible, because there's a clear understanding of data lineage, right? Like, “Oh, we capture in programmatic way the user intent. We capture where the data is going. We capture the purpose of why it's used by a specific system,” right? Obviously, Amazon can use your address to send you packages, but they're not allowed to use your address for crossing your data with something else, unless you’ve approved certain usage, right? Like if they're using it for

their purpose, they kind of have to specifically express it in the terms and conditions, right? Kind of the intent of collecting the data is important. And if the user opted out of certain things, they need to respect that. So it's kind of lineage is used in – It's a lot about understanding how those layers are laid out and how to make sure you know you could be looking at your KPIs at the company. And you want to understand how this metric is actually produced from the data that's collected upstream. Like understanding all those the years of transformation. It may be about understanding where your user private data is going and making sure you use it appropriately. It could be about making sure that when you push a change to one of those transformation, you're not breaking downstream analysis, or downstream dashboards. So lots of those different use cases for understanding lineage. It's really about having a Google Map for your data pipelines.

[00:46:30] JM: Okay. We got five minutes. We haven't really talked about Datakin. You've referred to open source projects, Marquez and Open Lineage. Marquez gathers metadata about a data ecosystem. Open Lineage gives you standardized lineage across your lineage and metadata. I can see how these two would be connected. I can also see how these two would be worth productizing. So give me the pitch for how Datakin utilizes these open source projects as quickly as possible.

[00:47:07] JLD: Yes. So Open Lineage is about standardizing the collection of lineage like across the data ecosystem. Anything you would be using, it would have lineage exposed in a standard way. Marquez is the reference implementation that allows you storing this lineage and versioning it and keeping track of all the changes. Datakin is a tool that builds on top of that, right? So the same data that is being collected through Open Lineage and Marquez you can use in Datakin to specifically improve your data pipelines operations, right? So this is about how do you trust your data? How do you make sure your data is updated timely in a reliable way? Right?

So Datakin is – We talked about a bunch of use cases, like compliance, operation and all of that. So Datakin is specifically focusing on how do I trust my data? First, it's building a map of the territory. So you understand your dependencies. What are my upstream dependencies if this dashboard is really important and there should be someone on call for it? Then every single dependency upstream, there should be someone on call for it. Like understanding all those

layers of transformation are really critical to make that very important dashboard or very important machine learning job always running.

The other one is really understanding upstream, downstream. When I change something, what other things potentially are impacted? The second level is to really be able to troubleshoot very quickly when there's a problem. If my data is late, or my data is wrong, being able to detect it, and very quickly troubleshoot the root cause of the problem. Because the main thing you need, once you know you have a data quality issue, or your data set is not being updated, or it's showing up later and later every day, is to understand lineage so that you can find the root cause. Like lineage is the key to understanding the root cause of the problem. When you have data quality issues, they come from upstream. Some other data set has changed. Or some other logic upstream has changed, which led to that data quality issue in the data set you really care about.

And so whether it's data showing up on time, or a data quality issue, you need lineage information to really quickly troubleshoot those problems. And today, it's really hard for people, right? It takes weeks sometimes to find the root cause of why we have a data quality issue. Where is this distribution skewed all suddenly? Or why are things getting slower over time? And so this is really one aspect we're doing, like building the map of the territory, understanding your dependencies, troubleshooting problem, finding the root cause really quickly. And third is obviously also preventing issues, right?

So from an operation perspective, there's lots of, in many ways, the service world is a lot more mature than the data world. Like people talk about, SLOs, mean time to recovery. How we track those metrics? And so it's very much about applying those same principles to the data world. How you make sure your data shows up on time and it's correct and you can trust it? And if there's a problem, you know that this is a problem, and it's not like you don't know when there's a problem or not. And really building trusting data and making it reliable. And whether it's for your KPIs and very important dashboards, or whether you're using it for machine learning and improving your product, or recommendation engine and all of that, things that treaty needs to be always working.

[00:50:44] JM: Okay, that sounds very appealing. I wish we had more time. Julien. It's always a pleasure. I recommend anybody who has these kinds of data problems to check out Datakin.

[00:50:53] JLD: Thank you.

[END]