EPISODE 1000

[INTRODUCTION]

[00:00:00] JM: A data platform contains all the data that a company has accumulated over the years. Across a data platform, there's a multitude of data sources; databases, a data lake, data warehouses, a distributed queue perhaps, like Kafka. There're also external data sources like Salesforce or Zendesk.

A user of the data platform often has a question that requires multiple data sources to answer. How does this user join multiple data sources from a data lake? How does this user join data across a transactional database and a data lake? How does the user join data from two different data warehouse technologies?

Presto is an open source tool originally developed at Facebook. Presto allows a user to query a data platform with a SQL statement, and that query gets parsed and executed across the data platform to read from any heterogeneous data sources. For some use cases, Presto might be replacing the technology Hadoop MapReduce, or perhaps Hive, the technology that is built on Hadoop MapReduce.

But for other cases, Presto is solving a problem in a completely novel way. Justin Borgman joins the show to discuss the motivation for Presto, the problem it solves and the architecture of Presto, and he also talks about the company that he started; Starburst Data, which sells and supports technologies built around Presto.

If you enjoyed today's show, you can find all of our past episodes about different data infrastructure subjects by going to software daily.com. You can search for the technologies or the companies mentioned, and if there is the subject that you want to hear covered, you can feel free to leave a comment on the episode or you can send us a tweet @Software_Daily, and you can also interact with us through our mobile apps which have all of our past episodes, and those are all available softwaredaily.com.

[SPONSOR MESSAGE]

[00:02:04] JM: As a programmer, you think an object. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers and the cloud area. Millions of developers use MongoDB to power the world's most innovative products and services, from crypto currency, to online gaming, IoT and more. Try Mongo DB today with Atlas, the global cloud database service that runs on AWS, Azure and Google Cloud. Configure, deploy and connect to your database in just a few minutes. Check it out at mongodb.com/atlas. That's mongodb.com/atlas.

Thank you to MongoDB for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:00] JM: Justin Borgman, welcome to Software Engineering Daily.

[00:03:03] JB: Thank you for having me.

[00:03:04] JM: You've been in the world of data engineering for about a decade. Give me your condensed view on how the data ecosystem has evolved since the days of Hadoop.

[00:03:13] JB: Sure. I think what's been really interesting is kind of the legacy of Hadoop, and a lot has been spoken about Hadoop's prime, perhaps waning at this point in terms of the number of people deploying it for new use cases.

But I think it actually brought about a lot of really important concepts that will live on likely for years and maybe even decades to come, namely this idea of a data lake. That's used to be exclusively synonymous with Hadoop itself, but now with the emergence of cloud and S3 and cloud storage in general becoming kind of the new data lake, a lot of those concepts live on just in a different form. I think that's a lot of the legacy. Including the development of open source data formats like ORC and Parquet which now have a renewed, I think, future with a cloud object storage as the new data lake.

[00:04:08] JM: You spent four years building Hadapt, which was a Hadoop company that was acquired by Teradata. What was the use case you were solving?

[00:04:15] JB: Back then, somewhat similar to what we're working on still today, which was essentially this idea building a SQL engine for the data lake. Back then, that was Hadoop. We were kind of SQL on Hadoop before it was even called SQL on Hadoop. Really, the first query engine for Hadoop to allow you to run fast interactive SQL analytics, and that really enable new use cases for users to do traditional data warehousing analytics style tasks, whether that's business intelligence or simply running ad hoc SQL analytics.

That was our mission. We had spun out of the Yale computer science department in 2010 to found that business, raised venture capital, built that over four years and then ultimately sold that business to Teradata.

[00:04:58] JM: When you were building Hadapt and then sold it, at the time the Hadoop vendors were building steam. AWS was constructing their own big data products, like EMR, and large enterprises were starting to adapt big data. How were those enterprise customers choosing between the different data providers when they're looking out over the Hadoop vendors, the cloud vendors, the classical data products? What were the choices that the enterprises were making?

[00:05:34] JB: Yeah, great question. I mean, I think first of all, there's always the question of cost as well as flexibility, and I think those two points in particular were really well articulated by the Hadoop vendors; Cloudera, Hortonworks, and perhaps to a slightly lesser extent, MapR in allowing people to basically store their data very cost-effectively, work with it at scale and really do analytics on "big data". I think that was part of the value proposition. I think that the challenges and maybe where Hadoop didn't fully hit the mark was on delivering high-performance SQL-based analytics.

I think when Hadoop was first starting to become more mainstream, some people at the time thought that SQL was a legacy language and that would be sort of the end of it and people would be learning how to use MapReduce and programming in MapReduce. I think that was sort of a miss early on that that was later corrected by creating these SQL engines and I think

started to create this interesting sort of choice for enterprise users, “Do I do my data warehousing analytics in Hadoop or do I do it in a traditional data warehouse?”

I think the low-hanging fruit during that period of time was essentially moving kind of ETL related workloads to the data lake where maybe you could perform these kind of batch-oriented SQL workloads on a lower cost footprint and then only load your higher value data into your traditional data warehouse, whether that's Teradata or Oracle Exadata or what have you. I think that was kind of an early reference architecture that you saw during that period.

[00:07:15] JM: Let's fast-forward to today. If we talk about the data ecosystem today, I can think of a few core abstractions. You have the OLTP databases, like Mongo or MySQL Postgres. You have a data lake that you can dump all your different kinds of datasets into. You have a distributed queue like Kafka, maybe it’s used for buffering data, and you have a data warehouse that data probably gets loaded into four OLAP queries.

Give mere overview of the different components of the modern data platform.

[00:07:51] JB: Yeah. I actually think you articulated it really well, and I think, to some extent, many of those are really essential. You’re always going to need your OLTP operational style system to serve your application, be it a web application or what have you. Kafka plays a really important role or other streaming technologies for sort of that real-time streaming data.

Then to your point, at the end of the day, you want to be able to analyze all of the data that you have more holistically, and that's essentially really the problem they were trying to solve today with Starburst and with Presto as the technology, is the ability to access all of your data where it lives and be able to create a holistic picture via a SQL query as supposed to the time consuming nature of ETL pipelines. That's not to say they we’re eliminating ETL pipelines, but rather giving you and an option to at least access many of these data sources via SQL query. In some cases, you even have multiple OLAP systems where you have silos of data in one or the other and you want to be able to join that together.

[00:08:57] JM: Tell me more about frictions or the difficulties of an ETL pipeline.

[00:09:01] JB: Sure. If you're a large enterprise, let's say a big bank or any complex enterprise that has a variety of different systems to work with, the first step is sort of figuring out what data do you want to move and how do you want to transform it. There's a lot of human cycles involved in that process having to figure out the right transformations to get your data in the right format in this ultimate eventual enterprise data warehouse concept. That's been kind of the methodology I would say for decades, is that all of these different data sources have to feed pipelines into one kind of monolithic system, and that's certainly what made Teradata very successful I think through the 80s and 90s was benefiting from being that “single source of truth”.

The challenge is each of those pipelines takes time both in terms of human effort, but also in terms of the time it takes to actually physically move that data, transform that data, and then you end up with data duplication. You have copies all over the place. The copies may not be perfectly in-sync. One person's view of the data may actually be out of date relative to another person's view of the data. I think those are some of the challenges with the existing model, but that’s sort of the status quo today.

[00:10:20] JM: We’ll get into Presto eventually, but to sheds more light on the standard data platform within a company, what is the purpose of the modern data warehouse abstraction?

[00:10:35] JB: Well, I think the idea is to have that single source of truth where you have one location that you trust to be the most accurate, up-to-date, in-depth view of the business itself. The theory is if I'm pulling in the right pieces of data from these various data sources into one place, I can give my analyst ultimately representing the business the tools with which to ask really any question that they’d like and understand their business more holistically that way. I think that's the idea of the abstraction. Whether that all truly lives into one database or is distributed across a few databases is perhaps a different question. But I think that's the basic idea.

[00:11:20] JM: There are variety of popular data warehousing tools, there’s Snowflake, there's Red Shift. Spark is used for applications that are similar to data warehousing. Tell me about the breakdown of the different data warehousing consumers.

[00:11:39] JB: Yeah. I think the Red Shift and Snowflake community of users is very much that business analyst type of persona. Someone who is perhaps using Tableau, or Looker, or even MicroStrategy, or Cognos, whatever BI tool that suits their fancy to ultimately ask questions of the businesses data. They generally have a more of a business orientation to them, but certainly understand data. I think in today's day and age are becoming more and more sort of data savvy in terms of the types of analytics that they'd like to do. But ultimately catering to someone who's likely using a BI tool or some kind of an analytical tool on top of that database system.

I think Spark by contrast is actually a little bit different. I think that generally caterers more to the sort of hardcore "data scientist" who is doing more advanced analytics or training machine learning models, doing all those fancy things we like to categorize as AI. Generally, it's a smaller group of individuals, but one with a very high degree of technical sophistication. So they feel comfortable perhaps programming in R, or Python, or Scala, or what have you to express the analytics that they want to perform. I think that's one of the fundamental differences between perhaps the two groups.

[00:13:00] JM: Presto allows for querying SQL across any kind of data source, and in order to understand why that is useful, it's worth going back in time a little bit. Hadoop pushed us away from big SQL databases and towards queries that looked across files. Explain why Hadoop pushed us away from SQL and towards files.

[00:13:25] JB: Well, I think the idea was first and foremost making it super scalable. Kind of one of the foundational elements of Hadoop was this Hadoop file system which was distributed in nature. Had built in replication to make sure that you had really good fault tolerance from a persistence perspective, meaning that you'd have three copies of the data spread out across this cluster and if one or two machines with that data died, you still had your data. You had a lot of confidence in the resiliency especially as you scale across hundreds or thousands of machines.

I think the afterthought for Hadoop was this notion that actually SQL at least as a language is still critically important to the enterprise and not only has it been that sort of lingua franca for 30+ years. It enables a broader set of users to be able to access that data. That's really where the emergence of ORC and Parquet and Avro and other file formats first sort of came to be, was

really trying to store data in a more efficient relational database-like format that was still an open format, meaning that multiple tools could actually access this data.

To go back to one year earlier questions around sort of Spark, or Red Shift, or Snowflake, Red Shift and Snowflake are much more traditional in this sense that you load the data into the proprietary format of that database and you can only access it from that database effectively. By contrast, if you're storing your data in ORC or Parquet, I picked those two just because they tend to be the most popular that we find for sort of high-performance analytics. Then you can read it from Spark, or Presto, or Hive, or a variety of different tools. So you get a lot of flexibility out of that, and we often find that you might have that data scientist group training machine learning models using Spark is accessing the same data that now a business analyst may access through Tableau using Presto, let's say.

[00:15:21] JM: Presto originated at Facebook. What were the use cases that Presto originally solved at Facebook?

[00:15:28] JB: Yeah. That's right. It was created by Martin Traverso, Dain Sundstrom, and David Phillips back in roughly 2012. Open sourced in 2013. The purpose at the time was essentially to create a faster Hive for the most part. Hive was also created at Facebook, interestingly enough. Facebook has been very visionary in this idea of sort of open data warehousing, if you will. But they needed a faster, more interactive solution because Hive was really better suited for those longer running batch-oriented queries.

They sought to start from scratch. One of the challenges with Hive was that the software was developed I think in a time if Facebook's history where maybe not all the best software engineering practices were in place at the time. So it was very hard to extend and really challenged in terms of its extensibility. Hortonworks put a lot of money and time into making it better with various execution engines that they swapped in and out over the years. But ultimately it had a lot of technical debt associated with it.

Presto was a clean slate starting from scratch, totally ground-up development initiated by these three guys and really designed to be as fast as possible. There are some fundamental architectural differences that they chose in order to get that performance. First and foremost,

queries are fully pipelined through memory and there is no notion of checkpointing, which Hive has.

So there's a tradeoff there of course. Hive has great query fault tolerance. If it's a five hour job and you have a node failure, it'll keep going because it keeps writing intermediate sets to disk. Presto, you'd have to restart that query. But of course we would say, "Well, the Presto query should never take five hours to begin with." It's sort of like a different approach to similar problems.

That was a big part of the motivation, was just strictly query performance. Then the other piece, which I think has really set Presto up really well for the future is that it inherently had this idea of storage compute separation built in, and I think that's something people didn't fully appreciate back in 2013 when it was open sourced. I think people thought, "Okay. Here's another SQL on Hadoop engine," but it's a sequel on anything engine. For Facebook, they have a massive MySQL deployment, sharded MySQL probably the largest in the world and they wanted to be able to access that in addition to a lot of the data that they had in their Hadoop environments. Having a tool that was flexible that wasn't tied to any one type of storage was really advantageous as well.

[00:17:57] JM: That idea of check pointing to disk and the fact that it slows down a query. If I understand correctly, the reason that – A checkpoint to disk basically means like I might write a query in Hive that does something like query this set of files across HDFS. Join it with this other dataset. Filter out these entries, and then you want checkpoint to disk and then do two or three other operations before you actually get to the final result. The reasons it's going to take something like five hours is because you have gigantic datasets, and we're talking 10 years ago, something like that, 7 years ago.

In contrast, an entirely in-memory model would pull everything into memory, but if it's all pulled into memory on a single machine or if it's pulled into memory across several machines and one of those machines fails, you might have to restart the entire job because memory is not persistent. Do I understand correctly the tradeoff between the in-memory model versus the checkpointing to disk model?

[00:19:08] JB: Yes, that's the essentially correct. Of course, Presto, like Hive, is an MPP engine. You can have your Presto cluster be as many nodes as you'd like to sort of increase the total amount of memory available to execute those queries. Very often, we see that a Presto cluster is its own distinct cluster, which may actually have very minimal disc capacity at all since it's not actually storing data, but is instead configured more in the direction of processing power and memory and then has a good network connect to your data source, which maybe HDFS in this case to execute those queries quickly.

Now, we later did build a spill to disk function as an option for queries that may exceed memory just so the query doesn't fail due to not enough memory. That is a flag you can turn on effectively. But, yes, the architectural tradeoffs I think you articulated really well.

[00:20:03] JM: When Presto got open sourced, it became adapted by other companies; Netflix, Airbnb, I think Uber, some other companies. How have the use cases that other companies are applying Presto to? How do those compare to what Facebook has used Presto for?

[00:20:22] JB: Yeah. That's a great question. I would say those companies that you mentioned. Who were the early adapters? You're absolutely right. Probably had somewhat similar use cases and that they are also thinking in a data lake-driven model where their "data warehouse" is essentially either Hadoop or S3. Some of those companies had already moved to the cloud, like Netflix, for example.

Those use cases were somewhat similar. Again where S3 or Hadoop becomes that storage layer for your data warehousing activities, the data being stored either in ORC or Parquet and pretty very much and even split between those companies that you mentioned in terms of their predisposition towards ORC or Parquet.

But I think what really changed was maybe around 2015 when we started to get involved after coming to Teradata. I mentioned in their earlier part about history, Hadapt was acquired by Teradata. It was really there that we met these three guys; Martin, Dain and David, and basically asked them, “Hey, could we become contributors to this project? We think it has great fundamentals, great bones to the infrastructure, but is maybe missing some enterprise features that more traditional enterprises would need. Things like security, LDAP integration, [inaudible

00:21:35] integration, wire encryption, etc., etc., full ANSI SQL coverage. At the time, it was it was a pretty good starting point on ANSI SQL coverage, but it wasn't complete. It didn't support every single piece of the syntax.

Those were some of the things that we started to invest in early and 2015 that I think started to broaden the use cases. In particular, we also built additional connectors. I mentioned how kind of MySQL and Hadoop or maybe two of the early important ones, and of course S3 and S3 compatible object storage, but we started to build out an advance – Other connectors, like Microsoft SQL Server, or Oracle, or Teradata, and that allowed I think the more complex enterprise, if you will, the Fortune 500 enterprise who has already all these different data silos and doesn't benefit from the ability to sort of start in the cloud as many of these leading Internet companies do.

It allowed them to also have in great use cases for accessing data across these different data sources. I could go into some details on some of those customer examples, but I think that's where the project started to expand beyond just the sort of Internet companies and start to take on a life within retail, healthcare, financial services, media, telecom, etc.

[SPONSOR MESSAGE]

[00:23:02] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:25:19] JM: There are companies that are pre-cloud and companies that are post-cloud. The companies that are post-cloud that are entirely in the cloud; Netflix, Airbnb, Uber. Well, Uber, not really because I think they have their own data centers now.

But there's a significant comparison between those kinds of companies that are post-cloud that have at least had the option to build entirely in the cloud and have built mostly in the cloud versus companies that have some heterogeneity or entirely on-prem. That impacts how their "data platform" has gotten built. What databases they use? What their, I guess, data lake looks like? That has some consequences to how they're going architecture or use Presto, which we can get to. But can you just tell me little bit about how the infrastructure of these different types of companies impacts their data platform.

[00:26:15] JB: Yeah. I think that's absolutely spot on. I would say in the case of the pre-cloud, I guess as you described it, maybe the enterprise that's been around a little bit longer and has some of that infrastructure already built out in an on-prem environment. One of their biggest challenges, really, data silos. If you take let's say just a big bank because tend to be very complex in nature, often times their businesses are built through M&A, merging with other banks

and rolling up other banks and new business lines. Now, with each of those, they inherit a new database of some kind.

Ultimately, they want to be able to create kind of that 360 view of the customer, but that's very challenging. It creates a lot of complex ETL pipelines internally to try to get the data that they need, and very often those are IT projects where the business is saying, “I want the answer to this question.” They don't know how to get it themselves. So this notion of self-services is something that is highly sought after, but rarely sort of realized today. They have to go to IT and say, “Hey, we need this data and that data,” and then IT has to sort of figure it out, and that could be weeks before they can get to the answer that they want.

There's a lot of complexity there and a lot of data silo related problems. By contrast, the other group, the sort of post-cloud that you described, has a real advantage and that they can try to centralize all of their data into one place. What they very quickly realize is that the more data they can keep in S3 in open data formats, the lower cost and the more flexibility they have sort of going forward. I think that's why you see a lot of these Internet companies use S3 or, again, whatever cloud is your favorite. It could be Google Cloud Storage. It could be Azure Data Lake. It doesn't matter what storage, but the idea being that that's the low-cost sort of storage tier and you’re storing your data in ORC or Parquet and that allows you to access that data from a variety of tools, keep it in a low cost place and have really a lot of runway for how you build out your architecture going forward. Definitely, two different patterns that we see.

Honestly, that's part of what gets me excited about the prospects for Presto because I think what makes it very unique is it really can cater to both groups. I think often people ask me, “Is Presto a data warehousing analytics tool, query engine for analytics, or is it a data virtualization tool?” It's really a little bit of both. Meaning, you can have data in different data silos today and use Presto as an abstraction above that to get access to that data to join across those data sources, but it also sets you up really well for the future where you might want all of that data in a cloud data lake type of model and access it there. Because it can do both of those, you can sort of build your business or build your BI infrastructure on top of one common SQL interface without having to have multiple databases for different things.

[00:29:08] JM: If I think about architectures that – If I'm just ignoring Presto, I can think about setting up data connectors to my S3. Setting up data connector to my MySQL database and pulling data from S3, pulling data from MySQL into a data warehouse. Pulling it into Red Shift or pulling into Snowflake, and then doing complex operations over those data warehouses.

Alternatively, the Presto, I could execute a SQL query that is aware of the schemas of the data in S3 or aware of the schemas in my MySQL database, and Presto has its own data connectors and it's going to execute those queries across workers and eventually give me an answer. How would those two patterns for getting an answer to a large query that is spread across these heterogeneous data sources, how would those two query paths compare to one another?

[00:30:12] JB: One thing we like to talk about, and this will answer your question, is this idea of kind of time to insight. Essentially, what is the total time it takes from the moment that I have the question to the moment that I have the answer? When you think about it that way, the time to insight includes the time required to basically get the data where you want it to be able to query it. I think what you're describing highlights this very well.

In the sort of traditional data warehousing model, you would have to factor in the time it takes to ETL that data into that sort of central EDW model. In our case, the time is exclusively that query time. In some of those cases, it's possible that Red Shift or Snowflake might be slightly faster in terms of executing that query, because they've done a good job, especially Snowflake, in terms of optimizing their storage format for their query performance.

Let's say a query on Snowflake takes five seconds. Maybe it takes seven seconds on Presto. But if you factor in the time required to sort of go through that ETL process and get that data loaded in and then also of course the opportunity cost of now that data is exclusively in snowflake. Those are the kinds of I think challenges or at least factors that you have to think about holistically when you're thinking about time to insight. That's essentially the difference. With Presto, it's all done through that specific query as supposed to doing that ETL process beforehand.

[00:31:43] JM: Time to insight, in this case, you basically mean there's actually going to be more engineering work that's going to go into like designing the ETL process. If you have to

design some ETL system for an ad hoc query rather than you already have Presto configured with all of the connectors necessary for an ad hoc query, it's going to be shorter “time to insight”.

[00:32:06] JB: That's exactly right, and there's maybe even one other factor worth mentioning as well, which is the self-service ability. What I mean by that is that ETL work that you described is likely the time of a data engineer required to get that data set up so that the analyst can ultimately ask the queries of the data. By contrast, if you've got all your connectors set up, you can theoretically make that available to your analyst to just run SQL queries against that data. You can create even views that actually hide where that data is specifically located, which data sources, and allow the analyst to basically ask those questions without the involvement necessarily of the data engineer so he or she can focus on perhaps something else.

[00:32:47] JM: Let's talk a little more about the architecture of Presto. I know that if I issue a query to Presto, it's going to be issued to coordinator. The coordinator is going to work with worker nodes to talk to the individual data sources and to break up the work of that query to distribute the work among the workers. Can you just walk me through the architecture of Presto in terms of a query?

[00:33:16] JB: Yeah, that's effectively correct. The coordinator is the one responsible for creating the query plan and distributing the work among the workers. The workers are the ones who are actually executing the various components of that query. Again, you'd have hopefully a good network between those Presto workers and whatever data source you happen to be connecting to, whether that is HDFS, S3, or a traditional database.

It will then push down predicates to those data sources, and depending on the particular data source, some of our connectors push down more than others. But essentially you're only pulling back into the Presto workers ideally what is necessary to compute the join, let's say. Ultimately, the query results get sent back to the user from Presto via ODBC or JDBC connectivity. There's also a REST API as well.

[00:34:11] JM: In order to issue a query to the data that's available across my Presto infrastructure, my Presto system has to have an understanding of the schema of the different datasets that across my organization. So if I have like a bunch of datasets in S3, I've got a

MySQL database over here. I got a Mongo database over here. I've got Postgres over here and I want to issue some complex Presto query that's going to do a join between my Mongo database and one of my S3 datasets. I need to have an understanding of the schema of all those datasets. To what extent this Presto know the schema of all the datasets across my organization?

[00:35:00] JB: Yeah, good question. This is an area where I'll say there is some roadmap development being done to make this easier. But today, we do rely on the user or you can again create views to sort of obfuscate some of these, but we would rely on the user, let's say, executing the ad hoc SQL query anyhow to know the schema of the underlying data sources, because essentially what Presto is doing is it's actually reaching out to the catalog of that particular system.

If it's Oracle, it's actually speaking to the Oracle catalog. If it's a Teradata, it's speaking to the Teradata catalog. For some people using AWS, we also have glue catalog connectivity. In the case of Hadoop, it would likely be the Hive catalog, the Hive amount of store. Presto itself isn't doing any necessarily re-conciliation across these various schemas. It's assuming that within the query, you know what you're looking for.

Essentially, the syntax without going into too much detail is sort of data source a.tablename, data source b.tablename. You're actually specifying what data source the data is coming from. A lot of that knowledge is therefore a prerequisite. Again, you can use a Vue to hide where that data actually is and sort of simplify that from the user perspective, and many of our customers and end users do that for their users. Again, this is an area of interest where there'll likely be more development along those lines as well.

[00:36:31] JM: If I'm integrating with Presto on day one, what is the process of getting my – I guess is it a data catalog kind of, like the catalog of all the different schemas that I could potentially query?

[00:36:47] JB: Yeah. It's all done through these connectors. I think going back to that notion of sort of storage compute separation, Presto itself basically doesn't do anything until you set up at

least one connector. The most common of course being the hive connector just because a lot of people have been accessing HDFS.

Interestingly, the Hive connector is a little bit of a misnomer because it's the same connector you would use to access S3 or other S3 compatible ObjectStore. Really, it should probably be called like the data lake connector. But that's one very common connector. Again, you would deploy any other connectors for various data sources. Within that, you actually point to the catalog, and that's how it knows where to go to access that data. It's really all about this connector model and very modular that way.

[00:37:34] JM: The hive – I guess, the like legacy Hive infrastructure. How much does that know about the details of my different datasets? I guess I'm just not super familiar with Hive.

[00:37:47] JB: Sure. The hive catalog, and this is where it can get a little confusing. There's Hive the query engine, and there's a Hive the catalog or the meta-store, and they're actually kind of distinct things. For example, even Cloudera's product, Impala, I believe uses the Hive meta-store, Drill, which was another SQL engine for Hadoop also uses the Hive meta-store, and Presto also uses the hive meta-store.

Now the good news with that, if you've already defined your tables, setup your DDL in the hive meta-store, it's totally rip and replace. You don't have to actually change anything. You just point Presto to that Hive meta-store now and hopefully now all of a sudden you have faster queries. There's no data transformation or data loading. No redefining of the table definition. That's the really nice thing. Migration path from somebody already using a data lake architecture to Presto is almost instantaneous for that reason.

[00:38:43] JM: The use case of Presto, is it for ad hoc querying, for dashboards, for machine learning applications? How do people actually use Presto?

[00:38:55] JB: Yeah, great question. Ad hoc, definitely. I would say reporting, definitely. Kind of any of your traditional sort of data warehousing style analytical query workloads, so a lot of BI tools, connectivity. Dashboarding, certainly. We have started to include caching into Presto as

well. We'll talk more about that probably in the coming weeks and months around those kinds of dashboarding use cases where you're accessing the same data frequently.

I would say machine learning is probably the one category I would, again, carve off more towards a Spark use case, and that's where we see really these two tools working together very commonly. Again, serving up different constituencies within a company, again, the data scientist versus maybe the analyst, but working off of the same files and working nicely within the same architecture.

[00:39:43] JM: Can you tell me more about the relationship between the coordinator and the worker nodes when a query gets issued? How does the parallelism work? For example, if I've got a single really, really large dataset in HDFS, there multiple workers that are pulling that data off of HDFS, or to the workers just – Or is there like an individual worker for each dataset? Just tell me more about the parallelism model.

[00:40:16] JB: Yeah. I will be honest, this is an area where it's certainly like our CTO or somebody else on the team can go into more detail.

[00:40:22] JM: Fair enough.

[00:40:22] JB: On exactly the query execution, but it is definitely highly parallelized even among that one dataset that you're working on.

[00:40:29] JM: Okay. Do you know much about like how the – I'm doing a join, for example, if one worker queries MySQL database, another worker queries my HDFS database. What happens during a join?

[00:40:45] JB: Yeah. Again, this is a case where the connectors themselves depending on the sophistication of the connector and they do vary across the sort of portfolio of connectors. But what Presto will certainly try to do is actually push down filtering to the data source that you're accessing so that your only pulling back the data necessary to actually execute that joints. Rather than pulling the entire dataset, for example, you're only pulling the necessary rows or columns or what have you to actually execute that join across those two data sources. That

makes it more efficient. Minimizes the network bandwidth and allows you to use your memory more efficiently to execute that query as quickly as possible.

[00:41:24] JM: Does the information about what filters or I guess predicate push downs, does that information get created in like the parsing and analyzing and the query planning step within the coordinator?

[00:41:42] JB: Yes. That's exactly. Subject to the capabilities of that connector, it will set up the appropriate query plan and push down as much as it can.

[00:41:54] JB: Got it.

[00:41:55] JM: I issue my query to the coordinator. Coordinator figures out what I specifically need to get from each worker, and then the worker pulls the necessary data via the data connector.

[00:42:09] JB: Exactly.

[00:42:10] JB: Presto is optimized to read these columnar formats, like Parquet. Those optimizations, are they something that's unique to Presto? Are there other systems that are able to read columnar files with as much performance or does Persto do something unique with its optimizations?

[00:42:28] JB: Certainly, Hive can read from those file formats as well. There are other SQL and Hadoop engines that can read from that. Again, that speaks to sort of the history and the legacy of where these data formats came from really during that Hadoop heyday, if you will. But I think, again, what makes Presto pretty unique is the level of optimization. In the case of ORC, that's used at Facebook. They've done extensive tuning to make that as fast as possible.

I believe it's Netflix that uses Parquet and certainly many others and they've have done extensive tuning to the Parquet reader as well. Both of those file formats, you really can't go wrong with Presto because there's been such a level of optimization and it all comes back to the architectural difference that we discussed in the beginning around really pipelining execution

through memory as supposed to taking it in stages and checkpointing it at every step along the way. That's what's able to deliver such fast performance.

[00:43:28] JM: Presto has a cost-based optimizer. What does that mean?

[00:43:31] JB: That means that the query plan takes into account the time cost associated with executing that plan and can therefore do various types of optimizations around join ordering and the sequence with which you execute that query plan to deliver the fastest level of performance. That's something we built a couple of years ago and was kind of a big step forward for Presto. Immediately gave maybe 10X performance boost overnight for queries in particular that have joins, and sort of the more complex the join, the more the CBO gets exercised and is able to deliver greater performance gains.

I'll say just at a high-level, I think like, directionally, these are the level of optimizations they think were very interested in continuing to pursue with Presto, because the more optimizations we can put in there, the more advanced Presto is as a data warehousing analytics kind of platform. These are optimizations that are not necessarily new to data warehousing analytics. Meaning that Teradata or Oracle has many of these optimizations, but they're new to an open source query engine, and I think that's what's so exciting, is really taking the storage computer separator model, but building in the same level of performance optimizations that these more traditional databases have.

[00:44:48] JM: Can you tell you more about your interaction with the potential enterprise customers or the enterprise customers the you have. What are the kinds of use cases, the problems, that they're looking to solve with Presto?

[00:45:06] JB: Sure. Yeah. They definitely fall into the two categories that we sort of touched on a little bit earlier. One being this idea of joining across two or more data sources. One nice example there would be Comcast. Comcast has a viewing behavior from the shows that you watch captured in a Hadoop data lake and also has billing data captured in a Teradata data warehousing system.

They'd like to join across these two different systems to be able to correlate your viewing behavior with how much you spend across all upsell, a whole variety of sort of business benefits to being able to do those types of joins. Traditionally, that would've required an ETL pipeline to sort of get the data from one to the other to do those joins. That's a nice example of Presto as this kind of abstraction or query fabric, I believe is the term that they use to describe the role that it plays in that architecture.

Then on the other hand, we have companies like FINRA, the Financial Industry Regulatory Authority who are actually a pre-cloud, I guess, company that became post-cloud pretty quickly and has sort of led the financial services industry in that transformation where they went from a number of on-prem traditional data warehousing solutions into AWS pretty aggressively and today leverage everything off of S3, and I believe it's ORC file formats and inquiry that with Presto.

That's more of that kind of open data warehousing analytics category of use cases. Those tend to be the two primary drivers, and we like to stitch those stories together by saying that Presto is great for that entire journey, and we find that all of our customers are somewhere on that spectrum. It's just a question of where they're starting.

[00:46:52] JM: The first use case, that's joining two datasets. The second one you described is you've just got a singular data source and it's just fast for that particular singular data source.

[00:47:02] JB: Exactly, and you've sort of standardized on some open data format. Again, whether that's ORD or parquet.

[00:47:10] JM: The use case of the join, as you said, basically your alternatives there aside from Presto are like a big Hive query where you're going to be checkpointing to disk and it's can be slower, or an ETL job, maybe ETL things into a data warehouse and then query off of the data warehouse. That's going to have some complexity penalty. Would you say that's the tradeoff there for that use case?

[00:47:37] JB: Yes. I would just add that you'd have to do ETL in either direction, because you'd have to get the datasets into one place and you're either going to move it all into Hadoop and

then you have Hive perhaps to do that, which Hive has some performance limitations, but also you had to do that ETL step just to get it into Hive, or you've moved it all into, let's say, Teradata, or the traditional data warehouse and again you have that ETL step, but now you also have the consequence of perhaps overtaxing an already taxed environment. Many of these traditional data warehouse just because of the cost involved are often running at peak capacity already. So adding more data to the equation is often challenging and of itself and may require buying another box, for example.

[00:48:21] JM: What is required to deploy Presto? When you're talking to these companies and a company like Comcast, what would be required for them to integrate with Presto? What would be the deployment path?

[00:48:36] JB: Yeah. First of all, I would say we'll start with maybe the easy parts, which are if you already have data in Hadoop or S3 and it's stored in these open data formats and you're using the Hive meta-store for your table definitions, that part is pretty seamless. You can just point presto to it and query.

Now you would want to make sure that you have resources available for a Presto cluster. So it is also an MPP environment, and like we talked about generally, you want to configure it with good CPU and memory and less so on disk because you're going to be pointing to some other data source. If that's an on-prem world, you want to make sure you have a presto cluster that's configured appropriately and, again, has a high-speed interconnect with your HDFS, perhaps, if that's Hadoop. If you're in the cloud, you're going to spin up machines to execute that Presto query.

Now, what's nice is because Presto doesn't store data, you can actually spin those machines right back down after you're done if you'd like, and we've even built in some auto scaling capabilities into the Starburst enterprise offering just to make that a little bit easier for you. But that's kind of the first step, I would say, is sort of connect it with what you already know and then start to work with the connectors that seem appropriate to you. If that's Teradata, if that's Oracle, if that's – We have the snowflake connector as well. Whatever data source you fancy, start to work with that and set up that connector and start to run queries. That's where we're always

happy as well. I mean, if there's questions you have or query tuning related questions, we're answering those all day long. So we're happy to be helpful.

[00:50:10] JM: Tell more about building the business. What is the strategy of Starburst today?

[00:50:19] JB: It's what would be classically known as an open core model, and what I mean by that is the core presto itself is open source. It's licensed under the Apache license and we're very committed to advancing that open core. We're extremely active members in the community. We represent actually the majority of the committers to the project, and that's an important aspect to our identity. In fact, just a few months ago, the original creators Martin, Dain and David that we spoke about earlier, joined Starburst and they are now part of the crew here as well and very involved in many of those community efforts. We host events over the course of the year all over the world. We've done them in India, in Singapore, In Japan and of course the East Coast, West Coast, and that's an important aspect of it.

The way that we make the business work, the way that we pay our salaries is by offering extra enterprise features around that. In particular, we have security capabilities that are not available in the open source, things like role-based access control. Being able to do row-level, column-level, data masking, those types of access control capabilities are generally very important to an enterprise.

We have auto scaling, which we mention for our cloud customers. That's important. We have a lot around manageability. We have a piece of our product called mission control, which makes it very easy to deploy and set up these various connectors and run your environment. Then of course there is the additional connectors, some of which we've actually turned into parallel connectors, which are something that you can only get from Starburst.

For example, if you want to be able to access data from Teradata or Snowflake very quickly, we have a parallel version of those connectors that delivers better query performance when pulling data from those data sources to execute your query.

Then the last piece of course is the 24 x 7 support that you get from the actual creators of the project. That's kind of the offering that we wrap around that core, and I think the way that we

think about those tradeoffs is we want to make sure that Presto itself is as great as it can be relating to performance and stability and so forth, then add these extra features that an enterprise would need to deploy in a real production kind of mission-critical scenario.

[00:52:37] JM: Now, I can imagine the ideal integration experience for me is like let's say I'm a bank and I've got like 100,000 engineers or something, crazy amount of engineer. I don't know how many engineers a big bank has, but a lot. I've got so many different datasets. Ideal situation would be I get integrated with Presto and Presto has a catalog of all of my datasets, but it's perfectly permission so that the only people who have access who can perform joins in the right places are the engineers who should have access to that data.

I can imagine that setting up and properly permissioning all of those different integrations would be time-consuming, and I'm sure that like, realistically, an integration doesn't stretch across the entire organization. Probably stretches across some subset of the organization. But I just like to get a picture for what it takes to integrate with a large customer, or like do they want their entire – All of their datasets within the organization indexed within the Presto data catalog or do they just have some specific subset of use cases that they're asking for help with?

[00:53:52] JB: It varies. I would say, very commonly, people start with a smaller piece of the puzzle, and that may be simply accessing data in Hadoop or S3 or even on-prem object storage is actually an emerging data lake format that we've seen where people are using IBM's Cleversafe product or Red Hat's Cepth or MinIO, which is a startup out here in the Bay Area that are all S3 compatible. So they look like S3.

Whatever the case may be, they start with something relatively contained and perhaps it's joining data in that data lake with one other data source as a starting point. But usually one of our biggest kind of champions within our customers tends to be the architect or architecturally-minded individual who is thinking about the big picture and how do we build this thing that can stand the test of time as we kind of rollout are our journey to whatever that new architecture looks for them, in many cases has some component of the cloud involved. That's where I think the flexibility of presto is very attractive. I use the term sometimes optionality as being a core part of the value proposition and that it is so flexible that those data sources can change, where the data lives can change. It could be on-prem, in the cloud. It could be some combination. It

could be multiple clouds. There's just so much flexibility that I think that is something that is generally well-appreciated and creates kind of a roadmap internally for many of these customers of how they'd like to roll Presto out overtime.

[SPONSOR MESSAGE]

[00:55:28] JM: Today's show is brought to you by Heroku, which has been my most frequently used cloud provider since I started as a software engineer. Heroku allows me to build and deploy my apps quickly without friction. Heroku's focus has always been on the developer experience, and working with data on the platform brings that same great experience. Heroku knows that you need fast access to data and insights so you can bring the most compelling and relevant apps to market.

Heroku's fully managed Postgres, Redis and Kafka data services help you get started faster and be more productive. Whether you're working with Postgres, or Apache Kafka, or Redis, and that means you can focus on building data-driven apps, not data infrastructure.

Visit softwareengineeringdaily.com/herokudata to learn about Heroku's managed data services. We build our own site, softwaredaily.com on Heroku, and as we scale, we will eventually need access to data services. I'm looking forward to taking advantage of Heroku's managed data services because I'm confident that they will be as easy to use as Heroku's core deployment and application management systems.

Visit softwareengineeringdaily.com/herokudata to find out more, and thanks to Heroku for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:57:00] JM: What's been your experience building the business thus far? How does building a data company today compare to building a data company – What was it? 8 years ago, 10 years ago? How is the market different?

[00:57:15] JB: Yeah. This has been, I'll be honest, a lot more fun. I'll tell you a couple of reasons why. I think we did things a little bit differently this time around. I think first of all, open source is just so important and I think that was one of the clear lessons we learned through the first business. We were proprietary at Hadapt, but we sold on top of an open source technology. Even though we were maybe a better or faster query engine then Hive or Impala, because those were free and open source, they just had much broader adaption.

I think one of the cool things about Presto is obviously it is open source. It has a real community associated with it and it's global in nature. We have customers all over the world. That's been really fun. It's almost like a social media apps, like once you create it, you don't know where it will go. But sometimes you hit on something and it starts to take on a life of its own. I feel like there's an element of that with Presto that's been a lot of fun.

The other big difference for us, which has been fun, is we actually didn't take venture capital right away. We bootstrap this business for the first two years and actually grew it as a cash flow positive profitable business. I say that not to brag about it, but rather that it actually gave us a lot of freedom to make decisions sort of with our own, I guess, level of sort of experimentation built-in. We could sort of be patient and take our time and do things the way that we felt needed to be done. I think that was a really refreshing way to build a business especially in those early days where you're sort of trying to find product market fit and understand use cases, understand how people are using it. We really didn't have any time pressures, and I think that's also special.

I think that's now – Even though we've raised capital, we just raised 22 million from Index and a gentleman named Mike Volpi, who's had a lot of success around open source businesses, in particular, Confluent, Elastic, etc. One of the nice things is I think we've kept that culture of essentially trying to be authentic in the way that we build this business and make sure that the business model is a real one. I think there's a tendency sometimes, particularly in Silicon Valley, to raise an absurd level of capital on the hopes that a business emerges at the end of the day, and I think because of the nature of how we built this business, we like to think that we're sort of focusing on the right things and want to build something sustainable that will actually stand the test of time, and I think that's where the business and the technology are really well-aligned.

The architectural decisions that Martin, Dain and David made even in the earliest days of Presto at Facebook were really with the foresight of thinking, “We want Presto to be a multi-decade project, something like a MySQL or Postgres that's been around for a really long time.” I think, similarly, as we sort a layout the architecture for Starburst, we’d like to see the same kind of thing and build a real sustainable business.

[01:00:03] JM: What were those first two years like building a profitable business? Was it mostly around consulting?

[01:00:09] JB: It started with consulting and support and then quickly graduated to that open core model with really the first enterprise feature being role-based access control. We introduced in maybe the summer or fall of 2018, and that was a critical feature for us I think to start to really convert to enterprise subscription contracts. That really allowed us to accelerate the business.

It was a ton of fun, like I said. It was also a ton of hard work, because we’re always understaffed by definition. I think to run a profitable business, you must have to be somewhat understaffed at least relative to the venture model. The first year was me selling, and then we had a couple of guys selling along with me and it was just sort of laying one brick at a time as we built the business.

Ultimately up to a little over 50 people I think when we first raised money in the fall. That was a lot of fun. Now it’s the next chapter, which we’re certainly accelerating that growth and we’ll quickly surpass a hundred people now as we continue to grow that business.

[01:01:10] JM: Why did it take so long for a Presto company to come into being?

[01:01:13] JB: That’s also a good question. I think a big part of it is that I think Presto was overlooked as a SQL on Hadoop solution. I think it was sort of mis-categorized that way. In a period of time where each of the large Hadoop vendors had their SQL engine of choice. They weren’t going to choose Presto because, let’s say, Hortonworks already had Hive. Cloudera has Impala, and MapR had Drill. I actually remember while I was at Teradata, I approached

Hortonworks and I said, “Hey, you guys should include Presto in your distribution.” They were like, “No. No. We’ve got Hive.”

I think that was just the mindset at the time. They’d all place their bets. So, Presto was sort of an afterthought. I think the other pieces that I think Presto got better and better and better a little bit under the radar. I don’t think a lot of people knew how much we were investing while we were at Teradata to really improve the engine. Lo and behold, a few years later, it’s actually really awesome solid technology. Now I think a lot of people are probably asking that question. But it sort of had the room to grow a little bit under the radar, which I think was good for it.

[01:02:19] JM: Do you have any benchmarks off the top of your head for how Presto performs compared to Hive?

[01:02:24] JB: Generally I would say at least 10X. There are benchmarks on our website and certainly ones that we could point you to. Generally, 10X plus in terms of performance.

[01:02:34] JM: Do you have any knowledge of why Hive didn’t go in the direction of being more memory-centric? I guess just the fault tolerance side of it, the reliability side of it. They were more interested in reliable batch processing.

[01:02:48] JB: I think that’s exactly right, because that’s what they were good at and what so many people deployed Hive for is sort of became, I guess, the chain around their leg on trying to advance or change that architecture.

Also, the codebase for Hive just became so complex overtime. I think the participation of so many different parties without much strict code quality governance around the project made it very hard to continue to extend. My hat is off to the Hortonworks team for everything that they did to improve it. But it was no small feat to even get it where it is today.

[01:03:27] JM: Yeah. That use case for like reliable nightly reporting, I guess you wouldn’t really want that to fail, and that was a pretty big Hive use case, right? People integrated pretty tightly and relied on pretty tightly.

[01:03:42] JB: Yeah. Also, I would say like batch-oriented ETL jobs. I think, again, going back to one of those early reference architectures. In the Hadoop world, a lot of it was take the data into Hadoop, prepare it and then pipe it into a Teradata or some other traditional data warehouse to do the analytics.

Many of those workloads were essentially being piped into something else, and I think people found that Hive just wasn't delivering enough of a query performance to actually do interactive queries against the data directly with Hive.

[01:04:13] JM: When you look at the potential for AWS to build a Presto – I mean, AWS does have a Presto offering, right? Presto on EMR thing.

[01:04:24] JB: Yup, that's right.

[01:04:26] JM: How does that feature functionality compare to what you've build thus far with Starburst?

[01:04:34] JB: Yeah, good question. In fact, I'll even plug AWS's other offering as well, which also competes with us, which is Athena. Athena is actually Presto under the covers. Not everybody knows that, but –

[01:04:43] JM: Oh. I didn't know of that.

[01:04:44] JB: Yeah. That's actually Presto under there. My answer will kind of apply to both of those, which is that I think this is where having the committers to the project is really critical. Not only can we ensure that our releases of Presto are the most stable, most up-to-date with features because we're driving the project forward. But also it allows us to make sure that our open core model of sort of extra enterprise features are features that AWS doesn't have.

Interestingly enough, we mentioned glue catalog is one of the catalogs that we support that's obviously an AWS data catalog. AWS didn't support statistics for the cost-based optimizer for Presto using the glue catalog. We did before they even sort of thought about it. I think you'll see continuously as we go forward here, we'll be consciously making sure that the enterprise

version that we put forward is well-differentiated relative to what you'll get on EMR. I think that's the important thing to understand about AWS's approach to open source in general, is they're generally just pulling a release from GitHub and essentially making it available.

Because of that, it's prone to stability issues. In many projects and certainly Presto is this way, the releases may come out every couple of weeks and nobody is necessarily regression testing those significantly in the community. These are areas where, again, Starburst can differentiate around performance, stability, the newest optimizations to the query optimizer that we talked about earlier, security, connectors, etc. Whereas they're sort of in the position of essentially just taking whatever we make available for them.

But we also think that anything that is good for Presto is ultimately good for us. So there's no doubt that there are many people using EMR Presto today or Athena today that we think of as potential future prospects for us going forward.

[01:06:42] JM: Given that that is the strategy, it's like basically the Amazonbasics.

[01:06:48] JB: Yeah. That's a great analogy.

[01:06:50] JM: It was on Basics Kafka, your Amazonbasics, MongoDB, or Amazonbasics Presto. Knockoff, I'll buy it. Hey, I'm wearing Amazonbasics jeans right now, or Amazon Essentials. I love it. It's all I need, right?

[01:07:06] JB: Sure.

[01:07:07] JM: I'm not going outside today. But from your perspective, why did people change their licenses? If this is all Amazon is doing, is making Amazonbasics Elasticsearch, why do companies feel compelled to change the licenses around open source projects to insulate themselves?

[01:07:32] JB: Yeah. Well, I think this is partially a consideration of what scale you're at as a business. I think the companies who have changed those open source licenses that you're

referring to, many of them, Mongo, for example, are now multi-billion dollar businesses where every little bit sort of counts.

We're still in grow the pie mode, I would say. For us, maybe less threatened by what particular market share that Amazon has. I don't think we have to have 100% today. I think it's more about growing that Presto market broadly speaking. At least that's the way I think about it at this stage. I think some of those other companies are trying to capture more of their own market and that's why they're thinking about it that way.

I'll say I guess in their defense, that Amazon does profit off of their labor. It becomes even personal at some level. I think whether it's the companies or the individual developers themselves have invested tons of time and money in building this project and yet Amazon monetizes it almost as well as they do. I think there's some personal element there as well.

[01:08:43] JM: Do you feel it's an erosion of open source norms or is it just a natural evolution?

[01:08:53] JB: I guess probably maybe no other time in history have we really had this kind of cloud model that maybe changes the script a little bit in terms of Amazon can essentially offer software for free, if it wants to, and still make money off of some margin on the infrastructure. That puts them in a very powerful, strategic position I think relative to all the markets that they're playing around AWS.

But I would also say that I think there is value in the kind of enterprise features and capabilities, and I think a lot of people particularly enterprise customers want to choose best of breed. I think that's where there will continue to be a market here for these. I think the doomsday scenario would be that none of these vendors make money anymore and therefore they can't even contribute to the open source. That would be very bad. But I don't see us anywhere close to that at this point. Yeah.

[01:09:50] JM: No. Probably not. I mean, it's hard to imagine a business big enough to be worth copying by Amazon, yet not big enough to subsist despite Amazon's pressure.

[01:10:06] JB: Right. I'll also say just one other point on that topic, is that AWS – I mean, we could spend hours just talking about AWS. I think this is an area where the other cloud vendors have chosen to differentiate as well by partnering with these same vendors that you're describing rather than competing with them and thinking of that as potentially a competitive advantage for those clouds. They're all competing with AWS as well. So rather than having their own flavors of these various open source projects have been much more keen to work with the vendors to provide those best of breed solutions. Perhaps even as first party offerings on those clouds, but really embracing the developers of the open source itself.

[01:10:49] JM: Although to a large extent, that doesn't matter that much today, because most of the people who are going through a cloud provider to get Elastic, or MongoDB, or Confluent, or whatever, they're desiring the integrated experience, and like if I want an integrated experience, it's probably because I want it on AWS. I mean, there are people who are on Google Cloud, I suppose, and maybe that would matter. But, I mean, I understand that the differentiating point of the cloud provider, it's interesting. But, yeah. I mean, does it matter to the market today? Does it matter to the customer today?

[01:11:26] JB: Well, I think there are some customers who are certainly persuaded by Azure or Google's approach to their offerings, and that they're I think both trying to offer premium products, essentially, better products to the market in terms of the capabilities built there. Some customers may make that choice. I think a lot of customers are also thinking about the cloud very holistically and where are the costs here, or the costs there.

But I think the other potential benefit for vendors is that many large customers are starting to sort of hedge their bets with respect to their various cloud vendors, right? There's also the multi-cloud approach. You can create a stack that is agnostic to any one cloud vendor. Then you've again you've created some flexibility or optionality for yourself. I think some of the larger, more sophisticated customers are thinking about that already today.

[01:12:16] JM: Last question. How will the data ecosystem look different in 5 years?

[01:12:20] JB: Yeah. I think first of all, you're going to see more and more shift to the cloud in terms of workloads, and I think this notion of storage compute separation will become more and

more kind of table stakes for data analytics. I think you're going to see that even the traditional databases, like a Teradata, for example, will increasingly embrace the storage compute separation. I think they're all hard at work on sort of how can we store the data separately from the compute so that we can be more cost efficient? Because it really comes down to sort of cost economics. I think that's another really big thing that the cloud movement has sort of brought to the table, is being able to manager your costs on an hour by hour basis.

Your infrastructure has to be able to allow you to have that efficiency, and that's where like autoscaling again as a feature I think used to be maybe a novel concept, but I think will increasingly become almost table stakes for any large enterprise to get the most of the infrastructure. That's part of what the cloud affords that on-prem doesn't, right? On-prem, you have to buy all the infrastructure upfront and you're paying for it whether you're using it or not. In the cloud world, that's not the case. If you can be very efficient with how you're using those resources, you can actually make a real difference for yourself. I think those kinds of features will become increasingly important for all of the database vendors in the space.

[01:13:42] JM: Justin, thanks for coming on the show.

[01:13:44] JB: My pleasure. It was a pleasure being here. Thanks.

[END OF INTERVIEW]

[01:13:55] JM: Being on-call is hard, but having the right tools for the job can make it easier. When you wake up in the middle of the night to troubleshoot the database, you should be able to have the database monitoring information right in front of you. When you're out to dinner and your phone buzzes because your entire application is down, you should be able to easily find out who pushed code most recently so that you can contact them and find out how to troubleshoot the issue.

VictorOps is a collaborative incident response tool. VictorOps brings your monitoring data and your collaboration tools into one place so that you can fix issues more quickly and reduce the pain of on-call. Go to victorops.com/sedaily and get a free t-shirt when you try out VictorOps. It's not just any t-shirt. It's an on-call shirt. When you're on-call, your tool should make the

experience as good as possible, and these tools include a comfortable t-shirt. If you visit victorops.com/sedaily and try out VictorOps, you can get that comfortable t-shirt.

VictorOps integrates with all of your services; Slack, Splunk, CloudWatch, DataDog, New Relic, and overtime, VictorOps improves and delivers more value to you through machine learning. If you want to hear about VictorOps works, you can listen to our episode with Chris Riley. VictorOps is a collaborative incident response tool, and you could learn more about it as well as get a free t-shirt when you check it out at victorops.com/sedaily.

Thanks for listening and thanks to VictorOps for being a sponsor.

[END]