EPISODE 1281

[INTRODUCTION]

[00:00:01] JM: The company Streamsets is enabling DataOps practices in today's enterprises. Streamsets is a data engineering platform designed to help engineers design, deploy and operate smart data pipelines. Streamsets data collector is a codeless solution for designing pipelines, triggering CDC operations and monitoring data in flight. Streamsets transformer uses Apache Spark to generate insights about data across multiple different platforms. Their control hub is a single hub for managing data pipelines, data processing jobs and execution engines.

In this episode, we talked to Arvind Prabhakar, CTO at Streamsets, about the history of data engineering and how we got to where we are today. Arvind is also an official member of the Forbes Technology Council and a member of many other projects on the Apache Software Foundation. He was previously Director of Engineering at Cloudera and a software architect at Informatica before that. I hope you enjoyed today's episode with Arvind Prabhakar.

Our first book is coming soon. *Move Fast* is a book about how Facebook builds software. It comes out July 6, and it's something we're pretty proud of. We've spent about two and a half years on this book. And it's been a great exploration of how one of the most successful companies in the world builds software. In the process of writing *Move Fast*, I was reinforced with regard to the idea that I want to build a software company. And I have a new idea that I'm starting to build. The difference between this company and the previous software companies that I've started is I need to let go of some of the responsibilities of Software Engineering Daily. We're going to be starting to transition to having more voices on Software Engineering Daily. And in the long run, I think this will be much better for the business, because we'll have a deeper, more diverse voice about what the world of software entails.

If you are interested in becoming a host, please email me, jeff@softwareengineeringdaily.com. This is a paid opportunity. And it's also a great opportunity for learning, and access, and growing your personal brand. Speaking of personal brand, we are starting a YouTube channel as well. We'll start to air choice interviews that we've done in-person at a studio. And these are highquality videos that we're going to be uploading to YouTube. And you can subscribe to those videos at YouTube and find the Software Daily YouTube channel.

Thank you for listening. Thank you for reading. I hope you check out Move Fast. And very soon, thanks for watching Software Daily.

[INTERVIEW]

[00:02:50] JM: Arvind, welcome to show.

[00:02:51] AP: Thank you very much, Jeffrey. Glad to be here.

[00:02:53] JM: You work at Streamsets. Streamsets was founded in 2014. And the world of data has come quite a long way. Data infrastructure has come quite a long way since then. I'd like to know about the initial thesis of Streamsets and the way that the company saw data infrastructure on day one.

[00:03:16] AP: I could go on forever answering that question, but I'll try to stay grounded and talk about something that most people can relate with. So think about a very simple example that most of us have gone through in our lifetimes, or some of us, who are a little elder than others, is this transition from film-based cameras to digital cameras. And the film-based cameras, that I grew up film-based cameras, and those were expensive, and they required a sufficient amount of planning. And you can't just like keep clicking pictures, because they were expensive, and they required time investment and getting the final output out. So consequently, all the pictures that you took as a family were well-planned, or so, or at least there was an intention to plan them well so that you kept the costs in check, as well as you got the product that you want it.

Nowadays, however, with the advent of digital cameras and every phone carrying like very superior high-end cameras, you take a trip with the family, you come back with thousands of pictures, and there is no way that you can actually curate very easily or very quickly all the specific nuggets that you want to string together in a memory about your trip. So, consequently, what ends up happening is you have this delusion of pictures, but then you end up relying on the very algorithms that Google Photos and Apple Photos provide for you to recognize places, people, the kind of landscape that you're dealing with, right? So these tools help you curate the necessary extract the necessary things that you need to string together a set of pictures to form the memory that you're looking for, that you want out of that experience.

But if you take a step back and see how this landscape has changed, what has really happened is you started off with a plan for getting some pictures. But now you've ended up with an after the fact processing to extract the pictures from a whole bunch of pictures that came your way. And that same inversion of focus has happened in the enterprise with data infrastructures. There was a time prior to 2010 when every bit of information, every piece of information that the enterprises worked with, was well thought out, designed, and kept in mind while the applications evolved around those. That funnel has inverted. Now, data is everywhere, and applications are popping to make sense of data. It's a very similar parallel.

So what are the tools you need to succeed in this world, the tools that the enterprises need to succeed in this world have to work with a mindset that these pictures already exist. You need to get to the right pictures so that you can form your memories. I hope that gives a little bit of a context on the kind of problem space that Streamsets is in. And this was really the epiphany that got us to say that the problems we solved for in data integration in the past two decades prior to 2010, they have come back, and they've come back in a very different manner that the old tools and old systems cannot solve for.

[00:06:25] JM: Okay. So you've described something pretty abstractly. I'd like to get a little bit deeper. So if we go back to 2014, data infrastructure in 2014 mostly meant, if I understand correctly, Hadoop pipelines, some data warehousing, some batch jobs. But compared to today, data engineering was totally in its infancy, where today you have Airflow and Spark jobs and a number of other streaming systems. You have much better data warehouses. You have data pipelines that feel more stream-oriented and continuous rather than batch. So if we take the world in 2014 versus the world of today, what are the products that Streamsets has built along that timeline?

[00:07:12] AP: So the core, the heart of our platform, is centered on this notion of data pipeline, smart data pipelines. Data pipelines, as you pointed out, Jeff, they existed. You could have

Flume pipelines, Kafka pipelines. Prior to that, Informatica and Talent pipelines, Ab Initio pipelines, you name it. Data pipelines have always existed. But what has changed and what Streamsets has brought to the market is this notion of smart data pipelines. The key difference between data pipelines and smart data pipelines is that smart data pipelines operate with intent-oriented information.

So let me take a moment to describe what that means. When you're moving data, and that's what a data pipeline does, it moves data from point A to point B, you have a choice. You can either build a design time pipeline, where you know exactly what the definition of A is, what the definition of that data is, what format it is, and what is its schema? What are its variances? What are its deviations? You know everything about that data source A, and you're moving it to data destination B. And you know that extremely well as well. You know what format it'll go in. You know what schema it will have. You know what fidelity it should have, and so on and so forth. So you create a design. You create a design, a mapping if you will, between A and B, and that becomes your data pipeline. Problem with this data pipeline is that if anything changes either on the source or the destination, or in the data itself, you now need to go back and reinvest that design time in order to get to that point of value where the mapping can be functional again. So that's one extreme.

The other extreme is what we call the opaque pipelines. Opaque pipelines are bytes in, bytes out. The pipeline does not care whether you're giving it good data or bad data. So Flume pipelines, for example, are opaque payload pipelines. Kafka pipelines are opaque payload pipelines, right? There's an interesting phenomena about people vacillate between one of these two types. So when you're working with Kafka, for example, some people will turn on Kafka schema registry. Guess what happens when you turn on the schema registry? It immediately makes it a completely design time pipeline.

So the problem with the opaque pipelines is that if there is a problem with the data or something changes in the source system, now all the consumers of that pipeline will have to deal with it independently. So you're kind of reinventing the wheel 17 times. You don't really know that that one problem can affect so many of your downstream consumers. The way this impacts the enterprises is even more subtle than that, because it's not very often that things are breaking in a really, really bad manner. More generally speaking, these are minor deviations that cause a rift in the very business concepts that this infrastructure is trying to support.

So, for example, a data dictionary, the definition of what a customer means to enterprise could be interpreted five different ways by five different downstream applications. And before you know it, you have a huge master data management problem downstream in the enterprise that has now taken roots. There is no easy way to back it up. So those kinds of problems are very much amplified in this modern world where the changes, the drift in the data is so persistent, it's so constant. So smart data pipelines actually find that middle ground where they say, “You can design a pipeline without knowing the full details of what the sources are or what the destinations are. But what you feed into the pipeline, the design of the pipeline, operates on the principle of intent.

So a very simple example is if I'm moving data that is sitting in S3 over to a table in my cloud warehouse, and all I'm required to do is to map a certain subset of attributes that are coming in from my S3, my object store dataset, then as long as those attributes exist in the data, the pipelines will continue to work. The source information can come up with more attributes. It can drop existing attributes, which are not part of the 17 things that you need, and pipelines will continue to work. There is no change in the design of the pipeline. There is no change in the implementation of the pipelines. The pipelines will continue to work. Of course, they will flag that something has changed. But based on the pipeline's smart nature, it knows that it has zero impact on the downstream applications. And it knows when one of those 17 things change, even in as slight and subtle a thing as just the precision of the number, the pipelines can flag it and identify that, “Hey, there is a downstream impact. A human intervention is necessary.” So that's the product. That’s the core idea that powers our DataOps platform, this notion of smart data pipelines. That's what we brought to market.

[00:12:11] JM: Can you get a little lower level? If I'm thinking about a data pipeline as having a number of different components, it's got some disparate data sources, files on S3, databases. And then you've got Spark jobs you need to run. You've got some sort of orchestration system. Can you give me a little more detail about what are the typical open source pieces or database pieces that fit into like a typical Streamsets customer?

[00:12:41] AP: Okay. So to answer that question, let me begin at a high-level, and I can go down – Interrupt and tell me if I'm going too shallow or too deep. We can tune it that way. So the highest level any data infrastructure, any data in motion infrastructure, has two independent planes or two independent dimensions. There is a control plane and then there's a data plane. Data plane is where data lives and moves. Control plane is what orchestrates and engages the data plane components to actually move the data.

Now, the questions of framing that you offer just now Jeff is very pertinent to the data plane, the sources, the destinations, the database technologies, the open source software, the orchestration engines and so on and so forth. These are all the moving parts mostly within the data plane system. And the data plane system itself has a deep concept of locality. This idea of where exactly the data is. So, for example, if the data is sitting on microkernel devices on IoT devices, it requires a different set of engines to operate to access the data. If the data is sitting on third-party systems, on object stores, which are not part of a managed cluster, it requires a different set of data plane engines. If it is sitting on clustered system, on distributed file systems, on aspects, on object stores that have access to clustered resources, it yet again requires like a Spark-like environment, a Yarn environment, where you can actually spin up the cluster and process it along the way.

So, broadly speaking, the data plane itself, as I said, is now divided into your edge devices, your IoT kind of platforms, your edge platforms outside of the cluster, the non-cluster resources if you will, and then on the clustered resources. And each one of these require a different handling engine in order for it to be able to make improvements or process it.

Now, within each one of these segments, the kind of sources and destinations you can work with vary by sources and destinations. I mean, the actual physical representation of the sources and destinations So, to give an example, if you're trying to get data out of an HTTP endpoint, a service endpoint, if you will, and move that data into a database, the physical manifestation of that service endpoint will dictate a significant amount of how your pipeline would work. Does it give you a single part access? A single invocation access to the payload that it's trying to send? Does it have multi part mime support? Does it use pagination? Is there a higher order application protocol that signals the end of a request? So you can go very, very deep into how specifically that HTTP endpoint works. It's not sufficient to say this is a REST endpoint, and it'll magically work. Because every use case, every enterprise, every integration, every system-to-system communication is different.

So where a smart data pipeline makes it so much more easier for the data engineers is it abstracts away a whole bunch of those lower level details. So what you see when you build these pipelines, regardless of what component, what bucket of data infrastructure it's going to work on, whether it's telling the edge device, in the IoT devices, whether it's on off-cluster or on-cluster, what you see is a canvas, and that canvas allows you to pop-in, mix and match your sources and destinations. And you draw arrows and you drop in boxes or stages in between them that allow you to modify, transform the data.

Now, if you're transforming data that is coming from non-clustered resources going into some other destination, you're obviously operating row-by-row, batch-by-batch. Modifying your ability to transform is limited to what can be done on a streaming pipeline. Whereas if you're on a clustered resource, whether it's a streaming cluster or a batch cluster, it doesn't really matter, your abilities are far more different. They're far more sophisticated. And they're more attuned to clustered resources. We often make the mistake, generally speaking, of saying that, "Hey, is clustered pipelines better? Are non-clustered pipelines better?" And the answer is they both serve different purposes. So they're useful for different use cases. And each one of them will operate very, very poorly if they're subjected to a wrong use case.

So clustered pipelines, for example, are great when you have pushdown compute that deals with large volumes of data. You can do joins. You can do heavy lifting transformations and so on and so forth. But they absolutely break down and become extremely expensive and bad for the purpose if you're trying to access physically tie down resources outside of the cluster. And the moment you do that, now, anybody could tell you that, "Look, you lost the value of running a cluster workload." So at that level, the pipeline's themselves are able to translate, whether it's a Spark application, or whether it's a standalone engine, or whether it's a microkernel engine, right? The pipeline's themselves can be pushed down to those three separate domains of the data plane.

If you step back from that, all of these pipelines that are moving data from one point to another or multiple different points to multiple different points often can be mapped together into a single topology that is then supporting a business process at the highest level. So what we do see at the control plane level is the orchestration of multiple of these pipelines that together then form the basis for you, your corporate IT, or your platform engineering teams, to then devise SLAs that can be applied end-to-end for large complicated workloads that have multiple pipelines, multiple hops, spanning into those systems.

[00:18:56] JM: Could you take me through a specific prototypical use case, or maybe a customer case study that stood out to you that will really illustrate what a typical application looks like?

[00:19:09] AP: Yeah. So we have the highest level few significant use cases that we see applied over and over again by some of our customers. One common one is hydrating data lakes. Data lakes are this concept that you have large volumes of data kind of organized loosely on top of a local storage that is then accessible to various component technologies and compute engines that can be layered to do analytics and higher order sort of value extraction processes. So hydrating data lakes requires you to move data not just from on-premises data sources, from like systems like databases, but also from systems that the enterprise may already be engaged, which a third-party systems like Salesforce, like Omniture and other sort of "HTTP endpoints".

So a prototypical use case would be hydrating a data layer that takes information coming in from some applications taking Kafka topics and moving them into the distributed store, the object store, the local storage, as I said. Or taking CDC feeds, chain data capture feeds from their on-premises databases or VPC databases, and then driving those into the data lake for further curation and processing. That would be one.

Cybersecurity is another. I would say cybersecurity, technically, if you take a closer look, is really a specialized form of data lake use case. It has the same problems as a normal data lake use case has, although it is far more specialized, and it has a time sensitivity to it. So it requires really high-grade of instrumentation and oversight so that you're not missing any key events that you should be getting in time to process and the assets that you find defend through cybersecurity initiatives.

CDC itself is a very important use case for us. A large number of applications that enterprises are building these days are downstream from various systems of record, whether they are relational or not, but they downstream the systems of record that operate directly on change data capture, because that's how they infer the state changes for the user-facing applications and then able to create events and process them in a manner that helps the business drive whatever the business processes these systems are supporting forward. So those would be data lakes, cybersecurity, CDC. Those would be the higher order use cases that come to mind.

[00:21:43] JM: So the change data capture use case, that's like, “I've got a Postgres database, and I'm telling the changed data feed,” or whatever it's called, “off of my Postgres database.” And I'm like triggering stuff based off of that?

[00:21:56] AP: Yes. I would say it depends upon what your downstream applications are. If you have only one downstream application, what you described here perfectly makes sense. More typically, what we see is the change data capture feeds actually do get democratized using like a broker or a messaging system like Kafka, that then gets propagated and used by multiple downstream consumers. That information in the lowest form of fidelity, or I'm sorry, in the highest form of fidelity can be used to recreate the databases state itself.

So if you think about somebody moving an on-premises database system in a manner that they want to replicate it to another database system, which is not the same identical vendor version, etc., they need a little bit of high-fidelity feed so that they can go create, run the same DDLs and the DMLs that are happening in the source system to recreate the state of the database. This problem becomes really tricky if you're going from traditional database system to a non-traditional database systems such that you're going from an on-premises, MySQL, and to not into an RDS, Aurora, MySQL, but you're going into, say, Redshift. Are you going into a hive installation that you're managing on a cloud property?

So the CDC workloads that we have support pretty much all of that heterogeneous integration between endpoints and applications, because we carry enough metadata that we can apply your CDC feed, which is typically used for replication purposes to much more sophisticated use cases where you can have an Oracle replicating into MySQL. You have a Mongo replicating into an Elastic. They all fall into the same category for us. I deviated from your question though.

[00:23:54] JM: No. No. That's fine. That's fine. I mean, in the change data capture, are they typically like throwing data on to Kafka as middleware? Or what are they doing with that change data that's coming raw off of Postgres?

[00:24:09] AP: Yeah, so we do see Kafka. We do see object stores. We do see Kinesis, Event Hubs, etc. Like brokers are prevalent. I would say nearly 60% of the cases, I would say there's some form of a distribution mechanism, whether it is a streaming distribution mechanism, like a message broker like Kafka or Kinesis, etc., or whether it's an object store that people are indexing that the downstream applications or directly accessing. That happens. I would say 40% of the use cases, there is an end-to-end connection. So you have the same pipeline depositing that feed into multiple destinations simultaneously. Those typically happen to be more time sensitive use cases.

[00:24:52] JM: Okay. If we think about the domain of data pipelines, again, Streamsets was started in 2014. I believe that Airflow came out in 2015, or maybe 2016. My understanding of Airflow is like, basically, it came out in in a world where everybody wanted something like airflow. And Airflow came out and it provided you sort of a scaffolding to put your data pipelines on and orchestrate your data pipelines. But it's been a great solution that lots of people have used, but it hasn't been as good as everybody's wanted. And now you have the rise of some newer data pipeline orchestrators, like Dagster, and Prefect, and some other ones. But I'd love to get your perspective on the open source data pipeline infrastructure space and how that compared to what you built out of Streamsets. Like when you look at the open source ecosystem, are they tackling the same problems as Streamsets was tackling or Streamsets is tackling? How does the open source data pipeline space compared to what you're working on?

[00:25:56] AP: So, I've been working with open source for a significant number of years, more than I would care to admit online. But the open source ethos, the ideas that power that innovation, are very relevant and very useful. Without open source, we wouldn't have like the Linux platform, for example. It's by far one of the most impactful platforms out there. But we know that Linux is open source. We can actually roll up our own distribution of Linux. We can get the Slackware kernel. We can piece together things from open source, like from some different repositories and so on so forth, and actually build out a platform and actually deploy it and use it. Nobody does that. And the reason why is because there are too many moving parts.

Open source is actually great for giving you those moving parts. And it's fantastic. Like Airflow is a great thing. And open source projects have their own set of “attach and detach cycles”. New technologies are constantly pushing for things that open source was not able to solve for the previous generation projects would not solve for. I don't necessarily look at that as an inherent sort of invalidation of what the previous technologies were doing. And I look at it as complementing use cases that emerge over a period of time. That doesn't mean that Airflow does not have a place. Airflow, in fact, is very, very useful. And so is Dagster. I would say they are two sides of the same coin. Enterprises are trying to solve for high-order problems, and these systems are very low-level, right? So there is the pain right there and then.

Now, one pain is assembling the Linux distribution for yourself. The other pain is in operating it. Assembling itself is a very intensive exercise. If you build a solution on open source, you take charge of all the moving parts, and you say, “I'm going to place them together in such a schematic that it’ll help me create the solution that I'm envisioning,” right? And you'll get to that point, that's fantastic. If that is low enough, an abstraction that's closer to the reality of the moving parts, then you're good to go. Chances are you don't need a vendor-supported solution. But if you're going really high-level, then you would be hit by the second order pain, which is when you put these things in production, things break. Things break not because they're bugs, because you didn't quite anticipate all the boundary conditions. You didn't quite anticipate all the changes.

This was in fact, the first realization that I had before starting Streamsets, because in my past life, I was working at Cloudera, where I interacted with many, many accounts. And one pattern that I saw consistently was that POCs would go great. If I want to create a report running off of a high warehouse, I can actually show that in really, really fast manner. And give me the data and I'll show you the reports and all the SQL queries that are necessary, the HQL queries that are necessary. The problem becomes when you actually go to production with that idea, the pipeline's that are feeding into that system are not sensitive to changes. So when your ipv4 changes into ipv6, your regular expressions do not automatically update themselves. Somebody has to go figure it out.

When the host header in the syslog message goes from all capital HOSD to capital H and small osd, there's a butterfly effect that changes the behavior of personalization of applications that being served out of the data center. These are the things which are very real today. And I feel like if you're operating at the ground level, open source is your best friend, because you have all the tools, you have all the knobs, all the dials, but you need to be close enough to the problem to understand in how to solve it. The higher order you go, if you don't want it to be your pain and your problem to figure out all the 70,000 moving parts that are going to support a migration to cloud, for example, which seems to be another use case that has emerged over the course of last year and a half as a very consistent driver for cloud adoption, if you want to operate at that level, doing open source will leave you kind of partially blind is what I would say, because you cannot anticipate fully in your current capacity at that level to see what are things that can go wrong. That's where you need a higher order abstraction like Streamsets.

[00:30:26] JM: The problems that Airflow was going after, are those the same kinds of problems that when you were going to market with Streamsets in 2014 that you were trying to address with customers? Am I right in that this is like a very similar problem space to what you were going after?

[00:30:44] AP: So we have a little bit nuanced viewpoint, and I'll try to explain. What Airflow solves is self-service for data pipelines, orchestrated pipelines, workflows. You could argue that something similar can be solved with, for example, Informatica and Talend. These are these are incumbents that have created mappings and infrastructure for moving data. You can do that. Why was Airflow preferable to Informatica and Talend? Because airflow is lightweight, it's open source, you can roll your own. You can you can get it to production really, really quickly. Can you operate Airflow at scale? No, you can, without sufficient scaffolding monitoring stack that you build out. Can you implement SLAs? Can you implement auditing, security, multifactor authentication, SAML integration? Can you do all of that? Yes, you can, if you're a developer and you are in love that technology, you can extend it. So there are lots of these things that surround it, which Informatica would provide out the box.

The problem with Informatica although is it's so hard to start with. You need an upfront investment of significant development resources, because, arguably so, it's a heavyweight platform. It's a full blown platform that you need to educate yourself and work with. Airflow, you can get up to speed in maybe less than an hour for most people, I would say, may not be at sort of the topnotch super power user level, but good enough to get started.

If you think about it, there's the enterprise grade nature of the solutions, and there is the frictionless nature of these solution. So I would say Airflow is more towards the frictionless, kind of roll your own self-service, get it going. Enterprise-grade, you have the incumbents, technologies like Informatica. What we solve for is bringing enterprise-grade nature to frictionless motion. That's where we're trying to come in, where we're saying that, “Look, you need productivity boost like Airflow can provide even more. You don't really need to roll your own security and authentication and auditing and all of that. We will give it to you in a very highly productive frictionless manner. But we want to do it in a manner that you can go production at scale. You can run your data centers on it. You can run hundreds and thousands and millions.” In fact, we have customers who run upwards of 5 million pipelines a day. And that's no small feat by any means. And then these are not shops with hundreds and thousands of data engineers. In fact, that project that I'm referring to was started out by six people, and they got it up to 1.2 million pipelines a day. And since then, it has taken on by itself. So that's the idea of bringing that frictionless to the enterprise-grade. That's where Streamsets fits in.

[00:33:32] JM: So that makes sense. I can very much imagine that being a really good starting point, especially in 2014, where people had enterprises –And enterprises still have today these kinds of problems. But back in 2014, it was really acute, and there was not widespread understanding of how to deal with this even if you wanted to roll your own. There wasn't really great guidance on that. It sounds like Streamsets was able to build a really strong go-to-market strategy around catering to that pre-Airflow, even in the midst of Airflow, fully managed data infrastructure data pipeline landscape. Now, how has that starting point, informed the products that you've built since the first product?

[00:34:22] AP: The way I would begin responding to that question, Jeff, is I don't think our fundamental platform definitions have changed over the years. We've been in business for seven years now. We're going to be seven years old very soon this month. The core ideas and the core thesis has not changed, has not shifted. If anything, we have now stronger proof points and a bigger result that this is really the problem we need to solve for. And we want other vendors to catch up and do things similar to how we're doing.

What has changed though is the shift towards the cloud. What has changed is in 2014, when we started, there a significantly more emphasis on on-premises solutions and on-premises data lakes and data infrastructures. We're seeing that wither away. We're seeing all of that consumed by cloud offerings and cloud services and so on so forth. That does not mean that the workloads and the kind of processing needs that enterprises have have changed. What it means is they're doing the same thing, but in a much more different sort of in a cloud-centric manner than ever before.

And there is an interesting side effect of that motion. That interesting side effect is that if you go back and look at what the enterprise infrastructures were built on in the 90s, in the 80s, in the 90s, they were appliances. They were these big refrigerator-like systems that would be plugged into a special sort of facility which had HVAC requirements and electrical requirements, and so on and so forth. The majority of the spend in those days was for the hardware that was optimized to run a specific software, which manage your data. That gave way to the early cloud, which was the virtualization of hardware completely. And when the virtualization of hardware happened, that meant from a capital expense to an operational expense. And in the process, the emphasis on the enterprise side shifted from buying hardware to buying software. And what we're seeing now is a further shift in the same direction, but applied to software. Software itself has gone from a capital expense to virtualized services that are being available.

So when I need to run a cluster, I actually run it using EMR, for example, or data proc. And I can spin up as big a cluster as I want. This is not necessarily just the hardware business. It’s not just the EC2 instances on which it's going to run. This is actually the specific version of Spark or the specific version of another computer infrastructure that is certified and supported by the vendor. This motion is what has accelerated over the course of last seven years, and especially in the last two years, where there has been lesser and lesser emphasis on building out on on-premises environment and moving more towards the cloud. And we're seeing it at top-tier enterprises. We're seeing it in banks. We're seeing it in healthcare, security industries. So this is very, very real. Our products over the course of last seven years have evolved. And in fact, I would say in the last year and a half have accelerated towards that cohesive cloud-first experience. So that's how we're evolving this market.

[00:37:42] JM: I want your macro perspective on data infrastructure. So one of the things that really surprised me over the last six years, when I started doing this podcast in, I think, 2015, I did a lot of coverage of these systems like Spark, and Flink, and Storm, all these kind of data streaming orchestration systems. And I think a lot of people assumed that the future of data infrastructure was going to be the you were going to have these kind of real-time processing systems doing a lot of work over time. And for a time, that was what happened. I think a lot of people are still using Flink and Beam and stuff. But my sense is that, for a lot of data infrastructure, the solution has actually become centered around the data warehouse, where you basically just throw all your data in the data warehouse, and then you do things on top of that. And in order to enrich the data warehouse experience, you do things with Kafka, and you do things with like Lambda functions, and you just trigger stuff. But a lot of the action just happens in data warehouse. Then you do like reverse ETL out of the data warehouse to do other stuff using Fivetran, or something, or Census, or whatever. That to me was a surprise. So I'd love to know your perspective on modern data infrastructure. And what has surprised you in the last seven years?

[00:39:03] AP: So what you described in terms of what has surprised you, Jeff, is I would say accurate. Nobody was talking about how important warehouses would be back in 2014. Yes, we always looked at enterprise data warehouses as part and parcel of enterprise data infrastructure, but with the shift towards the cloud and going forward with larger decentralized workloads that most enterprises are trying to orchestrate, it wasn't very clear whether the enterprise data warehouses will play the same role.

Now, we use the term warehouses still. We use the term data platform still kind of using them in the same concept with the same set of ideas as we used them before, like when talking about Teradata, for example, or Netezza, or Oracle, for example. But I think it's worthwhile pointing out that the underlying foundations of the system has changed. Today's enterprises have way more data under direct user management that has ever been the case before. Way more than ever before is the data sitting on object stores and distributed file systems and so on and so forth. And this data, not entirely all of it is cold in cold storage. In fact, the data lifecycle and the data management concepts are slowly evolving to catch up with this new reality of direct under management data.

So, while most of this evolution of warehouses, lakes, lake houses, name the abstraction that suits your model the best. While all of that has been very, very, you know, front and center in the evolution of data infrastructures, the real challenges that enterprises are dealing with, in my opinion, are what do you do? How do you redefine your data lifecycle? How do you manage your compliance workloads? How do you manage auditability across significantly replicated over and over, because it's so easy to store more data now and so cheap to store more data? There is almost a dysfunction of sorts in the enterprise where whatever you need you get. But then once your need is done, is addressed, you don't really know what to do with it. So there's this proliferation of management challenges around data that I feel, in my opinion, is where the world is headed towards, or that's going to come to head. Data is very valuable, but there is also liability. You need to treat data with respect in order to be able to get fast your own obligations and your own needs.

We've seen the evolution of like privacy requirements from a whole bunch of European commissions, and as well as in California, for example, that puts stringent controls and requires the enterprises to deal with these things in a very meaningful manner. But you can't really do that if your data is actually being shared across seven services, many of which are not under your control. How do you ensure all that comes together? So the part that I anticipate happening is clarity on what is the new way of lifecycle managing the data in this new environment of complex data supply chains, and technologies that will evolve to support those workloads in a manner that can be done at scale without requiring a corresponding scaling out of the human side or the manpower associated with managing the lifecycle of data. In other words, what I'm trying to say is the smart data pipeline that I see, as very relevant to embracing this diversity and heterogeneous environment complex data supply chain, will become more relevant in managing the entire data lifecycle over the coming years. I sure hope Streamsets plays a very important role. But I also know that we are not the be-all and end-all. There will be a whole slew of new technologies that will deal with master data management problem in a different way than it has been done before, that will deal with data dictionary and reconciliation and cataloguing in a different way that has been done before. Those things, I think the time is ripe, and in the next few years, two to three years, we will start seeing a lot of those innovations come to the market.

[00:43:19] JM: As a case study, why was Snowflake so successful? Why has Snowflake been so successful? Snowflake I think was another thing that surprised people. I mean, I guess the data warehouse categories surprised people, and Snowflake was the winner of the data warehouse category. What explains the rise of that company?

[00:43:38] AP: My humble opinion here is their relentless focus, their perseverance, and their commitment towards the cloud was a key differentiated aspect long before anybody else was beating those drums. And they kept on it. They kept to it. And over the course of the last decade, they've actually solved for the very compute challenges, cost challenges, that most of the newcomers in that landscape deal with. So they definitely were ahead of their times. And kudos to their vision and execution.

[00:44:10] JM: Yeah, I mean, they basically figured out that this category – Not only was the category important, but it has so much technical depth, that basically you can throw as many engineers as you want at it, and it's never going to stop being hard. You're never going to solve it fully. It's too hard. Are you a believer in this unified data lake data warehouse idea?

[00:44:37] AP: It depends upon how you define unified. And I know every vendor has a take on this. My point of view is if you take the vendors out of the equation, then the modern data infrastructure is highly heterogeneous. It has aspects of, strict, highly consistent ACID stores. It has aspects of object stores. It has aspects of distributed file systems and elastic storage and so on so forth. All those things serve a place. They serve a particular purpose. And they have a place in this modern infrastructure. Now, all of them are owned by one vendor. You say it's unified. If it is part and parcel done on one cloud provider, a little bit one data platform, a little bit – I don't see a problem with that either, as long as the tooling and infrastructure is sufficiently sophisticated to manage those diversities.

I think the cost equation, it's a little bit of the same play over again, but in a very different manner than what we saw in the 2000s when you would say like, “Okay, a typical enterprise only needs one ERP system, and one people management system, and one general analytics warehousing system.” And that's it. And you have these three things, and everything is done. And you can focus all your investments around that. Maybe we'll settle in into an equilibrium of that nature, with the key technologies that are existing today in the market, the cloud providers and the data platforms. But I'm also aware that each one of these platforms, each one of these vendors, is doing their best to not just serve the use cases that they serve the best, but also expand and income pass the adjacencies around them in order to be the one stop shop for the one enterprise. And then some good succeed. It's hard to delve from my perspective.

[00:46:30] JM: All right. Well, as we wind down, tell me about – Actually, I want to know the toughest technical challenge that you have encountered in the last seven years while building Streamsets.

[00:46:41] AP: Oh, boy. Toughest technical challenge. I will probably disappoint you with my answer, Jeff, but apologies for that. Technical challenges are not generally tough. It's the people challenges associated with the patterns and value recognition. This is what I struggle with the most. So, for example, when we started Streamsets, the corethesis, the core idea was this notion of data drift. I still believe it is as applicable. If anything, we talk about data ops as a new discipline. But if you really look at data ops, data ops is all about using smart pipelines to empower your organization to keep operating on a stable abstraction and stable foundation, which completely takes care of sort of the very fluid nature of the data supply chain underneath, right?

What it's trying to do really is trying to show how to deal with data drift at a mega scale, right? So I'm very much invested. Streamsset is very much invested in solving that problem every single day. And we have the tools, we have the technologies, we have the algorithms, we have the infrastructures and the engines to solve for that. But the people that resonate with that problem are the people who have been burned by that problem. I found it extremely hard to educate people who are not yet there. Look, this is coming towards you like a speeding train. You better tool up your infrastructure – And you don't have to use Streamsets. I mean, I don't get me wrong. This is not a shameless plug for Sstreamsets. It's the plug for the idea. What do you need to solve for? You need to solve for constant change. You cannot afford. Not every company is a Facebook that can have 500 people data engineering team solving for every single log injection problem. They can't do that. It's not scalable. Well, nobody is Google that can actually have a research wing trying to log down using machine learning to correct human errors, right? Google does that. We can’t do that. You can’t do that. Most companies can't.

So for us to succeed, for the broader world, for the non-Facebook's and the non-Google's to succeed, it's really important that we be mindful of what are our true challenges, and the true challenges are to deal with this constant drift of data, new schemas, and structures, and infrastructures in a manner that you can actually take full advantage of the technologies that are at your disposal today. For me, just this idea of helping a product owner appreciate how deep this is has been way more challenging than solving for it. So yeah, like I said, it’s probably not the answer you were looking for. But –

[00:49:25] JM: No. I think it's a good answer. It's honest. Cool. Well, Arvind, thank you so much for coming on the show. It's been a pleasure talking to you.

[00:49:31] AP: Thank you, Jeff. Appreciate it. The pleasure is all mine.

[END]