EPISODE 1457

[INTRODUCTION]

[00:00:00] JM: Data Integration infrastructure is not easy to build. Moving large amounts of data from one place to another has historically required developers to build ad hoc integration points to move data between SaaS services, and data lakes, and data warehouses. Today, there are dedicated systems and services for moving these large batches of data. Airbyte builds open source data integration systems. And the CEO of Airbyte, Michel Tricot, joins the show to discuss open source data integration and the engineering behind Airbyte.

[INTERVIEW]

[00:00:31] JM: Michel, welcome back to the show.

[00:00:33] MT: [inaudible 00:00:33]. Thank you for having me back.

[00:00:35] JM: First thing I wanted to discuss was all the progress you've made since we last spoke, and specifically, around building a cloud product. You've built an entire data integration platform, and it's been open source. And I think the last time we spoke, that was where you were at. And I think since then, you've built the cloud product. So I guess my first question is there's this chasm that every open source company has to cross where you go from being an open source product company to an open source product company with a cloud product. And I just like to get a sense for what was difficult in building up the cloud product.

[00:01:21] MT: Yeah. So when we started Airbyte initially, we always had the ambition of building a commercial product. And when we started it was really about, “Okay, what do we want to have to go in open source? And what is the value open source provide? And what a commercial offer would be?”

Now, what has happened is, in 2021, we’re not really thinking about building cloud yet. We're thinking of, “Okay, this is something for 2022. We're going to just focus on open source and just bring the community. And most importantly, growing the adoption for the product.”

And what we've realized six months after we released Airbyte in open source, it was around May, is we looked at how many people were downloading Airbyte. We looked at how many people were activating. So repeat getting data in Airbyte. And then we look at how many people were using Airbyte in production. And there was a drop between the two.

So what we did is we actually talked to the people who were churning. Bot really churning, because it would come back from time to time. So we just talked to these people and say, “Okay, why are you not using Airbyte in production? What is preventing you from adopting Airbyte?” And almost 100% of them said, “We love Airbyte. We want to continue to use it. We like that we can bring our own connectors on the platform that we can just dig into the connector’s code and fix it or just make it work for our use case.” That we don't want to operate Airbyte, because it's a data system. And we don't have the team to do it. We don't have the time. So that's when we decided to actually focus on cloud.

Now, we wanted to also make sure that as we create cloud, we continue to improve our potential because we're still a small team. And we’re wanting to make sure that by building cloud, we are not letting open source to be like going in the wrong direction, or to just not be as good. So what we did is we really build the product of cloud on top of open source so that every time we improve cloud, every time we improve open source, we improve the other.

And yeah, we'd say like the main change was just that we're a small team. So every time you make a choice like that, you need to think about, “Okay, what are you not doing?” So maybe we're investing a little bit less in some open source feature to do cloud. But at the same time, we structure the project in a way that improvement on one improve the other.

And now, when we released cloud in private beta in October, yeah, we very quickly got a lot of people signing up. And that's actually a very good signal for us that we have a lot of open source users or people that are interested in about just not want to just work with open source **[inaudible 00:04:13]**. So now we have like these two lines of product, I would say, and it's a matter of getting the team around it.

[00:04:21] JM: Can you describe the architecture of the cloud product?

[00:04:24] MT: Yep. So we very quickly invested in having Airbyte to run on Kubernetes. I think we had like a very, very alpha version. It was January 2021, it was barely working, I would say. But in June, we're really focused on scaling, I would say, like “infinity scaling” the number of connections that we can support.

And for cloud, we’re basically leveraging this Kubernetes deployment. And we've set a lot of rules are on how we partition the data movement within the Kubernetes cluster just to make sure that, yeah, data stays segregated by customers. But overall, it's a big Kubernetes cluster. And every single connection runs Icebergs. And we can just scale up and down.

And one nice advantage that we have with cloud is that in very careful at separating the control plane from the data plane. So control plane is everything related to configuration, scheduling, etc. And the data plane is more where the data is actually moving. And what we want to do at some point, and probably going to be happening in 2022, is having more Kubernetes cluster running across different regions, different cloud, so that the data plane can be actually something that you can decide and move wherever you want. So if you want to be running your data movement on Azure, boom! You just need to open a Kubernetes cluster on Azure. If you want one on AWS, AWS. And if you want to own the data plane and have it run within your infrastructure, just run your Kubernetes cluster, and we’d have a way to connect to it.

[00:06:12] JM: So each deployment of an Airbyte cloud – So if I spin up a nearby cloud for my company, that's an entirely independent Kubernetes cluster.

[00:06:21] MT: No, no, no, no. It's multitenant.

[00:06:24] JM: Okay, got it.

[00:06:25] MT: Like workspace within Airbyte are tied to customers. And at that point, behind the scenes, we have rules to make sure that pods run in isolation within Kubernetes.

[00:06:37] JM: So you've got a one large Kubernetes cluster. And, for example, there's a pod that handles all of the Salesforce ETL, for example?

[00:06:49] MT: No. It's going to be more. Okay, let's say your customer A, I'm customer B. You're replicating data from Salesforce to Snowflake. And every time the replication runs, we're going to spin up a pod that replicates Salesforce to Snowflake for you. If I'm doing the same, it will be running in a new pod. And every time the replication is done, we kill the pod. And next time it needs to be rescheduled, we reschedule that pod on Kubernetes.

[00:07:20] JM: So can you just describe if I connect to Airbyte cloud and I want to start doing ETL for all my different sources, can you just describe what is happening on the infrastructure level?

[00:07:36] MT: Yes. So if you want to start replicating data in the cloud?

[00:07:40] JM: Yes.

[00:07:41] MT: What will happen is if you want to go over the journey of the configuration to an actual replication – So first thing is you will connect your account. You will just set up the source, like provide us with the credentials for connecting to your Salesforce account. Create the destination, which is providing credentials for the Snowflake account. Then connect the two together. They set on a frequency.

And behind the scene, we have a Temporal scheduler actually running. And if you say, "Okay, Let's replicate data every 12 hours." Every 12 hours, it will spin up a pod that connects the Salesforce container to the snowflake container that are a part of that pod. And data flows between the two. But every connection is going to be a separate pod that gets **[inaudible 00:08:33].** I mean, you run it. Once it's done, it shuts down. And next time, like after 12 hours, same thing. And at that point, like the container that loads data from Salesforce would just the passing data to Snowflake.

[00:08:48] JM: Okay. Are there any interesting scheduler challenges to scaling up the large Kubernetes cluster that runs that whole Airbyte cloud?

[00:09:01] MT: I mean, the first thing is everything that – The way we package connectors is we package them as Docker image. And the reason we did that is because we want to be language

agnostic. And also, because Docker allows us to encode what is the environment that is needed for that connector to run.

So unless you have a Salesforce connector that require Python version, I don't know, 2.7 or 3.2, it doesn't matter to us. We don't need to install that environment. It's already part of your Docker image. And that's a very nice property of Airbyte, which is every single connector can run by itself. And you never have to worry about having anything else than just Docker install or like something that can run a container.

I mean, obviously, that's a little bit heavier than if we had like a lambda function to do it. But the lambda would be more complex to maintain over the long term, because you don't have this environment isolation. And some of the thing we had to be clever about is when we check connection, when we get the schema of the data at a specific source, these are UI type of interactions. So you need to reply very quickly. And scaling pods can take a long time. Like having them to run, giving you the result can take a long time. And here, we have to separate everything that we call like a user interaction result versus, yeah, a replication process to make sure that, yes, we have fast response so that we can display information on the UI, whereas the replication can take a lot longer. So here, it was about having separate node pools for the first interaction and like different Kubernetes configuration to have them like run faster, lower timers, etc., etc.

And for the scheduling, we're using a scheduler called Temporal, which is something that came out of Uber, that is known to really scale with massive number of jobs. And that has very, very fast scheduling functionalities and very nice guarantee in terms of how runs get reproduced, how jobs get retried. And we rely a lot on Temporal for like scheduling jobs on Kubernetes.

[00:11:22] JM: Can you describe in more detail what value Temporal adds?

[00:11:27] MT: Yeah. So first of all, it has a very nice SDK for Java. And a lot of the platform is built in Java. Containers are built in anything. But the platform is in Java. And it gives you reproducibility of the jobs that you run on the inputs, the outputs. It kind of resumes. And every single step – For example, if any steps in your Java code fails, or if something needs to stop, it will recover from the last instruction it was at. And that's a very good one for us. Because sometimes we have like advanced workflows on how we run outputs. And we want to make

sure that we don't do things that we've already done, or we don't – Like, yeah, we don't we run things that we've already run. And also, it's good for debugging ability, because now if a job is failing, we have access to all the inputs and the outputs. And it's easier for us to see what went wrong and try to reproduce the bug or the error.

And the SDK is very, very convenient, because it's completely – It's not like we were using Airflow, we would have to suddenly write scheduling in Python and have like all these jobs. It's part of our tech stack. And it scales to a lot of jobs. And that was a key for us.

[00:12:51] JM: We've done a few shows on done Temporal, maybe you could – I mean, this is going off on a bit of a deviation. But maybe you can just talk a little bit about the classic struggles of workflow management, since you've been working with various data workflows for years. Can you just talk about the struggles of workflow management historically and how those have eased up over time and how Temporal is something that that makes things a lot easier?

[00:13:20] MT: Yeah. So workflow management, the thing that is always hard whenever you start having workflows is to follow where your task is at. And anything is a workflow. I mean, any piece of code is workflow. It's just that when you start operating at the data level, or when you start having like multiple like disjoint technologies or disjoint steps, it's very hard to know where you're at. It's very hard to know the dependencies that are happening. For example, on our side, on Temporal, we need to check the connection before we do a sync. And we need to potentially pull the schema from our database. And these two things can be done in parallel. But you cannot stop the sync without getting both of this information.

And at that point, like workflow manager gives you an easy way to just paralyze. All these tasks can be paralyzed and ensure that you have a barrier somewhere to ensure that both tasks have been successful before going to the next step. And so you get all this dependency graph management for free. And also, you get all the tooling around it like. How do you look at what was the input? What was the output of each of these steps? And in terms of observability and debugging ability of your system? That is a huge help. Otherwise, in the past you would have just – What would you have done? You would have had like logs, and you will have had to put some breakpoints somewhere into your code to figure out where are things breaking, where are things not going well. You would also have to keep track of the input and the output. So persist them somewhere. And all this workflow managers, they do all of that for you. They track the

state. They make sure that each step is reproducible. And as an engineer, and as someone who works with data, this is key. You want to think of everything as almost like functions that takes inputs, provide output depend on each other. Because once you have that, it makes, yeah, the operation much simpler. I don't know if that was clear enough?

[00:15:24] JM: No, no, it makes a lot of sense. Is it just fundamentally like a challenge of competing over shared resources in like the workflows compete around shared resources, and therefore it's difficult to design workflows that are safe from problems like race conditions?

[00:15:44] MT: No, it's not just that. It's also the composability and composability of systems, where if you build one step to check connection, and you need to use it for another type of workload, you can reuse that step. So it allows a lot of your business logic to be composable and to have dependencies with each other and to be reused. So that's a big advantage of all these workflow managers, whether they are Airflow or Temporal, is this ability to compose what is the flow of your business logic. And this is at a high level. But that's ultimately what it does. And can just put all these building blocks, rearrange them the way you want, and just pass inputs and outputs from one test to the next.

[00:16:35] JM: So when it comes to designing a cloud product for something like ETL, there are all these distributed systems problems that used to be a lot harder to tackle, I would say, before Kubernetes, or before Temporal even, or even before Airflow. So given that you've been working in data integration for a pretty long time, are there any other points around what makes building a cloud product easier today than it might have been five or 10 years ago?

[00:17:14] MT: So first of all, having systems that Kubernetes is a game changer. With a cloud product, you never know. You never really know how demand is going to evolve. And Kubernetes gives you a lot of primitives around auto scaling your cluster. And also, the fact that the hardware you're running on basically doesn't matter. It's just you have access to CPU, and that's all. And this was not something we had before, which is every server that you had was important. And deciding where a job should run was a complex task. Whereas here, with technology like even **[inaudible 00:17:52]** is like that. Kubernetes is like that. It's just it's a scheduler, and it knows where things are running. What is the state of each node? And is going to take on itself the complexity of the scheduling. And I think that removes a lot of operational complexity from the team.

Also, because now you're not tied to a specific server, it also forces better engineering practices around having being stateless as much as possible, being idempotent as much as possible, because you never know if your pod is going to be rescheduled or something. It also forces you to think about having multiple versions of a pod running, like redundancy. So having a system like Kubernetes forces your team to think about problems that come as you become more successful as a product to other companies rather than stateless, and just auto scaling. Or as, yeah, in the past, every time you would have to think about, “Okay, how many servers do I need to buy?” That's not an issue anymore. Someone is simply to make the decision on the cloud. But as an engineer, you don't have to think about it anymore. It's just abstracted away from you.

One of the key thing that we think about when building Airbyte is I like to call it the iPhone button, which is a very simple way to get to your zero state, to be back to a state where everything works. And Kubernetes gives you that. And that's why also we’re making sure that each pod represents one connection that can be killed and put back. It ensures that as the connection runs, you don't keep bad state in memory or in your application. Just kill it all the time. And you restart it when you need it. So you don't have any kind of drift on, I don’t know, memory in terms of bad internal state. You always start from a fresh state. And that makes debugging and building software much easier.

[00:19:56] JM: So was the bulk of your code from the cloud product written Go Lang?

[00:20:01] MT: No. It's in Java. And what was the – I guess the reasoning behind that is, I guess, probably more important to have just like verbose, statically typed code rather than having – I mean, I guess that's what you would need rather than – I mean, you don't need anything particularly fast, like networking stuff that you might get out of Go Lang. I mean, any particular – Other benefits for building in Java? Just what you're familiar with?

[00:20:32] MT: I think it's more like what the team would send me always. And these are not places where we need to be – All the platform is not where we get the scale. We get the scale from the connectors. And we get the scale from Kubernetes. The platform is really about gluing configuration, gluing launcher, gluing Kubernetes cluster connect us together. It doesn't need to be the fastest. Also, to be clear, Java is also very fast. But it's also like sensitive code. I mean, things have to work. And having something that is statically typed is important. It makes

development easier in general. You don't need to just rely on test. You can rely on your IDE to know when you're doing the right refactoring. So I think it's just, yeah, a way of developing software that the team is not familiar with.

And historically, data, and big data, is very Java heavy, whether it's like – If you look at Kafka, if you look Hadoop, if you look at HDFS, if you get everything, or if you look at Spark. I mean, Java, Scala, whatever, but they come from a JVM.

[00:21:44] JM: So is there anything in the cloud product, in the architecture, that is like unfinished? The stuff that you're working on right now?

[00:21:57] MT: I think we're putting a lot of effort on the reliability of cloud. Our goal is to go in GA with cloud beginning of April. So we want to be ready for that. And at the moment, what we're doing is just doing a lot of – Like stress testing the system a lot to understand where is Temporal failing, where is Airbyte failing, and just try to just get that scale, because with the amount of people that we've had like signing up on the waitlist, we just want to be ready once we start announcing cloud in GA. So there are a lot of broken windows, and now it's just a matter of fixing them.

[00:22:37] JM: Where are the failures that occur that you're looking to make more reliable?

[00:22:44] MT: So sometimes we have pods that gets stuck. So figuring out what is the root cause? Fixing the root cause. It's not around scale. In general, it's just some very minor things on maybe tuning Kubernetes, tuning Temporal, or tuning our workflow. But these are very – These are things that don't appear a lot. They just appear from time to time, and we're just trying to figure out, “Okay, what is the root cause?” And fixing it forward.

But besides that, the system is pretty much ready. We want our users to have the best experience. We don't want them to have like stuck jobs or things like that. And another thing we're doing is just building the monitoring stack on top of it so that we know when job gets – If a job gets stuck, if data is not flowing fast enough, so that we can adjust, yeah, all the network policies and all of that.

[00:23:36] JM: When you think about the hardest parts of scaling the company, outside of just raw engineering problems, like what we've discussed with the cloud system, give me a sense of what's difficult about scaling a company when you've gone from – I don't know how many people you had last time we spoke. Probably like 20 or something.

[00:24:01] MT: Oh, no. Last time we took about – I think it was early last year in 2021. So we're probably like six or seven max.

[00:24:08] JM: Six or seven people. So now you have like 100, right?

[00:24:11] MT: No. We're about 50 people. Yeah.

[00:24:14] JM: 50. Okay. Yeah. So how have you changed organizationally over that period of time?

[00:24:21] MT: Yeah. So first of all, because we're a remote company, there are a lot of things that we need to make sure we also put in place to ensure that we still have some good glue between people. But scaling from seven, or five to 50, you need to communicate a lot more. I would say that's the first thing, is you have more chance that someone misunderstand something. So repeating what needs to be done. Like providing contexts and repeating contexts is key. And after that it's about how do you unblock people? Because when you grow at that pace, you need to make sure that people, when they joined Airbyte, they have a great experience. When you're seven, to 10, to 15, in general, it's a different profile of people. Like people who just are building everything, because nothing exists. But when you past 20 or 30, people also have an expectation on how they're going to be onboarding on the team to have an impact as quickly as possible? And that is something we've spent a lot of effort on, and we continue to spend effort, which is someone start at Airbyte, how do we give them the best experience? Yeah, I would say these are the – Like keeping a good future, keeping people happy, and just give them a good experience as they Airbyte is the key challenge.

Now, one of our goal is to grow to between 150 to 200 by the end of the year. And there will be other challenges. But investing in the people structure is key for us, because that is what will allow us to have people who feel good at Airbyte, people who have a good experience. And it will ensure that we also continue to keep the culture that we've built so far. It will create a bit of

chaos. But if we have the right people infrastructure, we can absorb that chaos and just make use it to our advantage.

[00:26:28] JM: There are a lot of newer data products that have come out over the last four or five years, or come to prominence over the last four or five years. And a lot of them are related to ETL or reverse ETL. And then there's also products like Segment, or RudderStack, or mParticle, these various data transformation, data integration API's. And that's kind of different than how things were 10 years ago, where you had to do everything with Hadoop, or just writing ad hoc ETL jobs. Although you do move to a place where you have this plethora of tools and plethora of places to manage. So obviously, it's easier in the sense that you don't have to manage as many pieces of infrastructure yourself. You have a lot more cloud hosted products that you can use. But when you go to this place of having so many tools that you take off the shelf, does it create any new problems for people that are using a wide variety of these data integration tools? Are there are there new problems that arise when it goes from this world of roll your own data infrastructure to use a variety of cloud managed products?

[00:27:59] MT: Yeah. I think every time you fix an existing problem, you generally create a new category of problem down the line. If you look at what existed 10 years ago, 15 years ago, these were all very monolithic solution, where it's very top-down adoption is generally the CTO or some exec, make the decision that, “Tomorrow, we're going to be using solution X.” And every piece of data is going to flow through that solution.

And you realize that when you have a monolith like that on end-to-end solution, in general, they've been thought with covering a prospect. They are biased in how the vendor was thinking about their processing. The moment you start going a little bit away from how that end-to-end solution were thought to be to work, you have to do something on the side. You have to take your team and build a secondary system to handle that. If you want to add more quality step into it, you can't, because these systems are not flexible to just build another block to control data.

And I think what we've seen over the past 10 years is that all these solutions are specializing to become much better at what they do. I mean, if you just look at data warehouses, that's exactly a symptom of what has happened, which is, instead of focusing on data quality, instead of focusing on data loading, instead of focusing on all of that, they just say, “You know what? We're just going to focus on putting superfast SQL on top of data. And we'll take on us how we

want to organize the data. How we want to optimize storing the data." But they just focus on the value they were providing. And they just wanted to focus on the compute. The problem is that once you're in that state, then you need to replicate all the other component that existed in this entrance solution.

And that is what has happened. So yes, it comes with – And that's why you have solution like Airbyte, your solution like DBT, you have solution like Tableau, like Looker, you have like all these – Or like RiverCTL, like Hightouch, all these little things that add themselves to create a stack that is the mirror of what your organization is. Just that nobody knows better than your organization. What you want to do with the data and how you want to leverage it.

And at that point, yes, you create a little bit more complexity, because now you need all these tools to integrate with each other. But at the same time, now companies have the choice to just add the tool that they want and to use the one that is going to be the best for their business. So if you need Tableau, you use Tableau. If you need Looker, you use Looker. If you need Hightouch, you use Hightouch. But you don't need to buy a full end-to-end solution that tried to do everything.

You can make a parallel was what happened in software development before. Everything was a monolithic application, then it became microservices. But you have two side now, which is monolithic – Some people think monolithic is better. Some people think microservices are better. In the end, it's never one or the other. It's just a compromise that you need to do to serve your business interests and your organization interests. And it's the same for data. Maybe for some places, you will need a monolith end-to-end solution. And for other, you would need just building rugs that communicating with each other that are providing value. I hope that answer the question.

[00:31:37] JM: No, it does. So there are overlapping pieces of functionality. Like there are obviously competing products in the ETL space, in the reverse ETL, or ELT space. And, obviously you have your bias. But I just like to get a sense for if we want to present this in an unbiased way, what are the axes on which people should be considering the different ETL and reverse ETL options? Or more generally, if I'm trying to construct my data stack, what are the questions that I should be considering?

[00:32:18] MT: Yeah. So I think there are axes to think about. There is also the skeleton you need to think about, which is you're looking at building your data stack, there are at least four pieces you need to decide on before you can start having like incremental additional building blocks. First one is you need to pick what is your processing engine. So that is the first step. You don't have a data stack if you don't have a processing engine. And the processing engine is going to be a data warehouse, a data lake house, whatever you want to call it. The thing that allows you to process data is the compute. Then you need to be thinking about how do you actually get data in?

So here, it's everything that is about extracting data from different sources. And that can be DMS, if you're on AWS. That can be Airbyte. That can be Fivetran. It can be many other players. It doesn't matter the vendor. I mean, obviously, I'm biased. I think Aibyte is better. But it's just about you have data. You have silos of data. Just find a solution to bring data into your compute engine. That's the first two steps you need to make.

And then you need to look at what happens on the right side. And what happened on the right side is – Well, now that you have there, what do you want to do with it? And in general, that will depend on what teams and people you have available in your company. If you have mostly non-technical users, but just BI the users, then you probably want to go for like a BI tool.

If you have a heavy gross team or marketing team, maybe you need to do some reverse ETL. It's just the dimensions are basically what your team is comfortable with. If you have people are technical, maybe they're good with SQL. Maybe they're good with DBT. They just having all these building blocks actually allows you to pick what is best for the team you have available or for the team you want to hire for.

And also, it can also be about what kind of guarantee you want to provide on data. Some people don't care about the quality of some data set, so they don't need to buy a quality tool. Some make very important financial decision based on the data that they put in. So at that point, they need to bring some quality into it. And that's why I like not having a monolith end-to-end solution is important. It's just you are building a better stack based on what your team needs and what talents you have available. But you always need the first two steps, which is you need your processing engine and you need downloading enginge. And after that, you decide how you want to leverage it.

[00:35:06] JM: So given that you're interacting with lots of large companies that are using Airbyte, are there any specific canonical challenges that you see companies with large data infrastructure encountering? Like, what are the modern canonical data problems?

[00:35:24] MT: So first of all, there are many. But to go from left to right, first of all, it's just the amount of places where they have data. That's a key problem for them. And today, many of them are just building connectors internal. That is one piece. The other one is going to be around – Especially as the stack becomes bigger, it's more around the discoverability of that data. So they have bigger and bigger team to extract insight from data or to do something with the data. But they don't always know what data is available and what it means.

So at that point, it is going to be more around like the cataloging and the discoverability of your data. And then you have a lot around observability. And here, it's more how much can I trust the data that I'm working with? Is it fresh? Are there any bugs there in my data? Am I making the decision was the right data? So just access. Am I accessing in the right way? And I'm actually accessing it? Can I empower more people in my team to do something with it?

[00:36:25] JM: So when you say access, are you saying like there's a struggle around properly permissioning large cardinality datasets?

[00:36:34] MT: So you have around permissioning. Like, should that team have access to that data? I mean, if you have social security numbers to your data warehouse, you probably want to make sure that only specific people have access to it. But it's more that with the simplicity of data warehouses, now companies can hire profiles and talent that are maybe a little less technical than a data engineer, but that have very strong domain experts in what they are analyzing. And these people, you can grow these teams faster than a data team. So you want to make sure that this team can be onboarded very quickly. That they understand what data is or manipulating. That they have visibility into what is the data that they're working with. So that's what I mean by discoverability and access. So permissioning, and just, yeah, the discoverability of it.

[00:37:27] JM: When you talk to companies that have existing data infrastructure, like a lot of – Let's say they have a big Cloudera installation of Hadoop stuff, or they have a lot of legacy ETL

jobs that they've written in ad hoc infrastructure, is it a priority for them to migrate to better ETL solutions? Or do they keep their existing ETL and just kind of put new jobs onto something like Airbyte?

[00:38:04] MT: I think it really depends. We have people who are just looking to completely move away. I don't think they're incompatible with each other. You can still have like a large Cloudera deployment, especially if you have like big operational workflows processing lots of data. And maybe you just want to have something that is faster for your analytics. And at that point, there is a bridge that you can have between your big HDFS Hadoop cluster into a Snowflake. And it's a different person that is using either of the system. If you're looking at Cloudera, it's going to be called data engineers doing something. Sometimes they provide interface in front of it. But if you're looking at data warehouses, it's more like analytics engineer, data analysts, etc.

So I think you can have a bridge between the two. It's just what do you want to do with that data? I've used both. I know we were using Hadoop and Spark for big joins across terabytes and terabytes of data. We probably could have done it with the warehouse wouldn't have been as cost efficient. I don't know. But what I know is that we were able to also feel a lot of data into data warehouses and let other people out there that were not data engineer to actually be very fluent with the data and to make the right decision with it. So it doesn't have to be a migration. I think these two systems can live together.

Now, the problem that both of them have is like accessing the data when it's across many, many sources. That's always the same point. It's just your team use more products, your team use more services, your team have data and more and more as for your GCP buckets, and just how do you get access to that and how do you empower people to access that data? But these are just compute engine if you think about it. So they are part of like the central place where once you have the data in, you can enable your team to do what they want with it.

[00:40:07] JM: As you look forward – The core product is getting very dialed in, and you have an opportunity to lever up into new products, what are your priorities for what kinds of new products to build?

[00:40:25] MT: I don't know if I would call that a new product, but it's more new ways of using our Airbyte. The thing is, right now we're really focusing on trading all the physical pipes between sources and destinations. That's the core focus of the company. And what we call our North Star for Airbyte is the reliability and the ubiquity of the space. So that is what we're going to spend a lot of our time in the future, is making sure that the pipes work, and that we have the right processes in place in case they stop working. And the second one is making sure that we can access data in more and more places. And more and more places can mean more API's more file formats, more than databases. But it can also mean more geographies. So whether it's having access to data in Europe, whether like having access to data in APAC. Like having data within your own infrastructure. And that's what we're going to be focusing on. So I don't know if you would call that new products. But the way we see it is just about reliability and ubiquity.

[00:41:38] JM: Is there something particularly hard about what you said about like accessing data in a different geo?

[00:41:44] MT: Yeah. I mean, you have a lot of local regulation. So at that point, you become a partner with your customers on what they can and cannot do with data. Yeah, like different geo is complicated, especially if you look at between Europe and the US, there are some regulation in place on what you can and cannot do with data. Same thing like if you have like medical data in the US, you cannot just move it without thinking about it. So what kind of security you need to put in place to ensure you're moving the right data?

[00:42:18] JM: Cool. Well, Michel, it's been a real pleasure talking to you. Time flew by. And congrats on all the success.

[00:42:25] MT: Yeah, thank you very much, Jeff.

[END]