EPISODE 1227

[INTRODUCTION]

[00:00:00] JM: ELT, or extract, load and transform, is the process that modern data pipelines use to replicate data from a source and load it into a target system such as a cloud data warehouse. ELT is a more flexible evolution of the traditional extract, transform and load workflow used in pre-cloud systems. The power of ELT relies on flexible integrations between data sources and their targets called connectors. The wide variety of data sources available to a cloud application today means that an ELT platform needs to handle a vast and growing set of use cases for its connectors.

Airbyte is an open source ELT platform built with the long tail of integrations in mind. Airbyte is secure, extensible and simple to set up. Developers can use Airbyte's platform to build the connectors they need for their specific use case without worrying about scheduling, orchestrating or monitoring. Michel Tricot and John Lafleur are the co-founders of Airbyte. Previously, Michel was the head of integrations at LiveRamp building data ingestion connectors, while John was the co-founder of StreamNation, Anaxi and CEO of CodinGame. Michel and John joined the show today to talk about why ELT has changed the way organizations manage and store their data, the technical challenges that exist in the world of data integration, and how Airbyte can give your infrastructure superpowers.

[INTERVIEW]

[00:01:20] JM: Guys, welcome to the show.

[00:01:22] JL: Hi, Jeff.

[00:01:23] MT: Hi, Jeff. Great to be here.

[00:01:26] JM: For any company that's working with multiple data sources and data integration, there's always been this ETL step; extract, transform, load. And today there's a shift towards ELT or even EL without T. So extract and load with no transform. Can you give an overview of just a brief history of ETL and where we are today?

[00:01:52] MT: Sure. So ETL is basically following the step of extract, transform and load. Meaning that all the transformation happened before the data is made available, which means that when you have people that actually require to leverage that data, they have to work with the data that has been transformed. So if for any reason the data was not transformed in the way they wanted or if they were missing data, they have to go back, change the transformation and re-ingest the whole data again. So this is very inefficient especially in today's world where we have a more tech savvy and analytic savvy audiences and where warehouses are becoming more usable by less technical users. At that point you want to give them as much autonomy as they can to actually extract insight from that data, and that's where ELT comes into play, which is you extract and you load and you stay as close as possible to the source of the data and you let the responsibility of actually deciding how the data should be transformed and what it should look like to the end consumer. And I think that's why we're seeing this trend today of removing the T and moving it as far away and as close to the consumer as possible.

[00:03:15] JM: So an EL job simply concerns sources and destinations. For an average company, can you give an example of what a source and a destination would be? How many data sources does a company generally use for an EL job?

[00:03:29] MT: Yeah. So on the source side it can be any place where you have data, and we call them silos of data. Meaning that they are very isolated. So it can be your SaaS service that your marketing team is using. It can be your security SaaS tool that one of your team is also using. It can be a cloud service. It can be a database. It can be a file. It can be any place where you have data that is stored and that you cannot leverage with other pieces of data. And a destination would be actually where you want to disallow that data, and that could be a warehouse, that could be a file, that could be a database. It can be anything that an end user can actually look at. It could even be a spreadsheet for what it's worth. And at that point, like you extract from all these sources, put it into this destination.

I would say in two days' work, we have a very fragmented market of tools. So a company use – I mean, I don't have like exact number of how many sources people are using. It depends on the stage of your company, but you can think of it as like 10, 20, 30, 50 number of different tools and at that point you want to – What you want with EL is to actually enable all these sources to be just centralized into one central place so that you can have this whole view with all the data into one single place.

[00:04:56] JM: What are the essential features of a data integration tool?

[00:05:01] MT: So essential feature would be around the variety of places where you can actually take the data from and ingest the data. Addressing the long tail whether it's like internal data sources or external data sources. So this is extremely important especially as you're building your analytics pipeline or your data pipelines. You want to get as much data as you can. The other one will be around understanding what the data is. Meaning what data schema, what fields are, so that downstream consumer can actually make the best choices on how to extract insight from this data. Then you also have data quality. Ensuring that any change at the source level, you're going to detect it, because you don't want to be extracting inside from bad data or you will be taking the wrong decision. And, John, will you add anything else?

[00:06:00] JL: I would add like data quality privacy compliance, but those are more enterprise features I would say. But, yes, that tend to be in there.

[00:06:10] MT: Yeah. To go on what John was saying is like making the data available to more and more profiles across an organization, you need to ensure that you have strict rules on who can access and who can do what with that data. So, yeah, that's an important piece.

[00:06:29] JM: What are the current limitations of existing data integration tools? If you think about the common tools like Fivetran or Matillion, what are the limitations of those tools?

[00:06:42] MT: I would say one of the most important – I mean, two most important limitation is, one, the variety of sources they can pull data from. Meaning that you are limited by a subset of all the places where you have data into this platform. The second one will be more around – More for FIvetran than for Matillion, because Matillion methylene can be deployed on-prem, but Fivetran requires you to actually let a third-party vendor to actually access your data. So, in general, within larger organization or mid-market companies or companies that have some security requirements and/or concerns, then they will never be able to just feed all that data to a third-party. They will want to have that internally and they end up prevailing the systems in that case.

[00:07:33] JL: I would add a third one. When we started Airbyte, we managed to discuss with 45 of Fivetran's [inaudible 00:07:40] and Matillion's customers, and they were always building their own connectors on the side either because it was not supported, but also because sometimes it wasn't supported in the way they needed. So there's this flexibility part that is missing with closed source tools like Fivetran and Matillion. And also being close source means that the go to market is really top-down. Before, a data engineering team can use them and test them. They need some kind of approval. While with Airbyte, as it's open source, we see just like a frictionless adoption that just tests us without having approval at any point. And so we hope that the way ELT is being done will change with open source. Thanks to this open source approach.

[00:08:33] JM: There's a long tail of sources and connectors. Can you talk about the niche sources and connectors that are out there and why this wide variety of long tail niche sources is relevant to what you guys are doing with Airbyte?

[00:08:51] JL: So we see – It's a 20%, 80% percent rule. The priority rule is still there, and we see that all the current platforms are focusing on that. But definitely the niches, there are quite a few. You have all the healthcare, all the fintech, all the international connectors like Brazil's equivalent to Stripe kind of things that are not covered by any tools right now. And there're also a lot of the enterprise tools, enterprise database that are not covered now.

[00:09:24] MT: And also like the existing ELT are very focused on customer data or like customer SaaS application, and we think that there is a lot more than just syncing customer data. You have like security data. You have audit data. The variety of data is not covered by what exists today, and there is this other like 80-20 percent that will apply for this type of connectors.

[00:09:52] JM: Why hasn't there been a widely used open source solution to ELT in the past? Has there been something lacking from the open source offerings on the market in the past?

[00:10:03] MT: I would say it's also a matter of like the market maturity and like the saviness of users around data with like Redshift was and like BigQuery and Snowflakes became extremely popular a few years ago, enable this new role. And at that point when you can enable less technical and more data savvy people to actually leverage data, they will have more need for this data. So that's something that was previously like the realm of data engineers or engineers. Now we're moving to like less engineer profile. And so that's why there is this need for more connectors. And at that point you have had some initiative in the past. I think talent started this way, but more recently you had Stitch Data who released Singer, which was this open source framework for building connectors. But in the end, like Singer, for example, didn't really take up especially after like Talend acquired Stitch. They kind of dropped the ball on the project and that ended up like just dying by themselves not being maintained. Not having like an entity that would support that particular standard. And I think today the market is mature enough. People are educated enough around data that there is an opportunity for creating that open source ecosystem.

[00:11:30] JL: I would add on a single question. I would add that we think that they did a few mistakes. So for instance, they didn't standardize that whole how connectors were to be built. And you ended up with a lot of different one repo per connector, per tap, that was contributed by the community. And without any standardization, maintenance is kind of difficult. So when you have nobody that is – No entity supporting the community, you see that 70% of their target, of the taps now, are becoming out of date. So that's why we're doing things differently at Airbyte. We have for the moment only one repo trying to standardize how things are being built so that maintenance can be just easier for anybody and made by anybody in the community.

And also one last thing that is pretty important, Singer didn't work out of the box. So you needed like a few hours and even one more full day to be able to sync data using Singer, while in a few minutes Airbyte lets you do that. You can just sync, replicate Salesforce or Snowflake within two to five minutes. So it enables us to address a large audience that can also contribute.

[00:12:55] MT: Yeah. One thing that makes the success of open source projects is generally the ease of use and the ease of onboarding, which isn't something that has been addressed with past initiative.

[00:13:08] JM: How did you guys start to work on Airbyte? What were the previous experiences that led you to focus on building it?

[00:13:15] MT: So we actually both had experience with building data system. So I started my career actually on financial data the end of 2007. And in 2011 I moved in the U.S. and actually started in this small company back in the day called LiveRamp, which is like a middleware for customer data allowing people to sync data across their database and across all the marktech and adtech ecosystem. And over there, to actually power the LiveRamp system, we had to build thousands of connectors. So I was a director of engineering and I was head of integration over there. So I had to really build the infrastructure, the process that are necessary to manage connectors at scale. Yeah, John, I'll let you talk about your own experience.

[00:14:10] JL: Yeah. And I would add that you were moving hundreds of terabytes per day. So you know how to really scale this. And my side, that's my fourth startup, but my third one was built on top of ETL pipelines. It was a platform on top of all your engineering tools. So you can just manage your projects from one dashboard, one interface. And we spent months, I believe, almost a year building these pipelines. It was so painful. So yeah, we both had that in mind, commoditizing designed databases.

[00:14:45] MT: In the end it's a type of plumbing that every single company is doing in isolation today, and we believe that we should be spending time on building the actual future of products more than the plumbing, and that's why commoditizing plays an important role here.

[00:15:06] JM: Talk a little bit more about what Airbyte does and what your vision for it is.

[00:15:13] MT: Yeah. So there are basically two aspects to Airbyte. On one side you have the connector protocol, you have like building and enabling the community to address the long tail of integration and solving their data replication issue. And on the other side you also have the core platform, which allows you to quickly get data pipelines up and running. What we want to do with Airbyte is really focus on, at least today, the community edition and making sure that every single feature that beneficiary to a single user stays and stays free for them and then go for more like a commercialization offering, which is more like an open core and like an open core model that will address the needs of the more like enterprise or larger teams around like data security, etc. But really, like the vision is really removing the burden from engineers to have to build and maintain this data pipeline while at the same time empowering these less technical users or less data, like data engineer profile to actually leverage that data. So that's why the way we’re building Airbyte is we have both like an API to configure Airbyte, but we also have a UI where like a less technical profile can start syncing data and start leveraging it directly from Airbyte.

[00:16:51] JL: I would add that – So with the open core business model, we're addressing the internal analytics use case, but also another use case that we can address is empowering our customers to offer connectors on their own platform to their own clients. So it's really empowering any data movement, whether it's just internally or also empowering your clients to move the data on your platform. And this would be then through the API. But we are focusing on 2021 on the open source part of the technology really and we focus on monetization only once we've become the open source standard to move data. So most probably later than ‘21 or ‘22.

[00:17:42] JM: Give a little bit more perspective on how Airbyte fits into the competitive landscape. We talked a little bit about how it compares to like Fivetran and Matillion earlier, but what are the main features that set it apart from its competitors?

[00:17:56] JL: We see different use cases. The first ones that we see with the companies that are using us is they want to replace fast channel stage. So they're using us like to have a Salesforce to Snowflake typically use this kind of things. And right now most of them are waiting for the license to renew before they switch to us. So it's more about testing in that case until the license. They won't renew the license. The other use cases is about all the connectors that they don't have using those platforms, and it's really about lowering the maintenance costs.

I would say that there's a third use case, which is like they're about to build connectors or they're building connectors on their platform, and we see that a lot as well more and more in the past few weeks. So this is the kind of three use cases that we see most.

[00:18:55] MT: Yeah. And all about what differentiators is, the way of adoption for Airbyte, which is we are open source and we have a very bottom-up approach, which is we let people – Like the people who are going to be using it on the daily basis or maintaining it on a daily basis to actually just take the code, run it, see what value it gets them and then start to use it. And from there we can expand to more use case. Maybe they will start with just, “Well, I just need this simple database replication. Let's try with Airbyte. Oh! It works. That's amazing. I’m going to use Airbyte for this particular use case. And now – Oh! Someone in my organization is asking me for like syncing data from, I don't know, like any kind of SaaS service like Salesforce. And instead of going there, building it myself or like trying to explore other vendors where I will have to go through red tapes and an approval because suddenly the data is being exfiltrated to a third-party I’m just going to see, “Oh, Airbyte supports it. They have a connector for Salesforce. Let's get the data connected.” And boom! It's done. So it's really about the ease of use and the ease of adoption directly from the people who are usually in charge of implementing these data pipelines.

[00:20:15] JL: Yeah. We indeed see a lot of adoption for one connector and then we grew filters alongside that we build the connectors for them. So it's exactly this, yeah.

[00:20:29] JM: Let's begin to talk about the engineering. So Airbyte connectors are simply Docker containers. Can you explain why you made this decision and how it fits into the architecture?

[00:20:39] MT: Yes. So one of the issue with existing open source project is building a connector. In general, okay, you need to read the API documentation. You implement a few functions to pull the data. And you can put that on GitHub, but beside you, almost nobody else will be able to run it. And what running it as Docker containers enables us to do is to actually also encode not just the code and the logic. It also allows us to encode the environment of how the connector can run, it needs to run. And also by having it as a Docker container it makes it so that we can very easily integrate its connector into our testing framework, because the difficulty with building connectors is not building it. It's maintenance. It's like the long-term cost of updating your connectors, fixing bugs. And it's like this thousand per packet problem where every time every user is going to discover one thing that you haven't really taken into account in your initial implementation and you're going to encode that into your code. And having them as standardized as Docker containers enables us to just fit this connector into any kind of testing framework and we can start validating that connectors are working and that like the tests are all passing. And if they are not, then how we can like notify either the community or like our internal team to actually fix it.

And how it's run also, the fact that these Docker containers enables us to scale faster, because now we can directly run these containers on Kubernetes. That's something we're actually working on at the moment. They can run in isolation. If we need more than one we can just create new pods. But it's really always the same things, like the ease of use, ease of development as well. Today, everybody who knows how to use Docker, they don't want to know which version of Python they need to install. They just put that into a container and it works out of the box.

[00:22:45] JM: And how would a developer write and plug in their own connector?

[00:22:49] MT: So we have a few parts of our documentation. We explain how to build a connector from scratch. We also provide templates for basic use case like an API use case or like the Python base container or connector. And in general the only thing that you need to do is run a few commands that will generate a scaffold of what a connector looks like and then it's mostly – For SaaS services, it's mostly like fixing all the fix miss in the cloud and making sure that you output the data in the proper format, and after that you can build your connector locally, you can test it, and at the end you can push the container and anyone who wants to use this particular connector can just pull the image and they have access to the connectivity. Like all the Docker images have a very specific interface on how you interact with them. So it allows us to automate all of that.

[00:23:57] JM: There is an extensive effort in writing a custom connector. So for every company, the use case and the functionality of a connector can be different. How will you ensure that custom connectors that people build across the community have standardization and have extensive functionality?

[00:24:15] MT: So that's a good question, and actually that ties back to what you were asking before about ELT versus ETL, is when you do ETL, you're basically encoding your business logic before loading the data into your destination. With ELT, it allows you to actually be less opinionated about what data you need to extract and like how it needs to be presented for the end user. And that is I believe why ELT is becoming so popular. Is that suddenly you only need to write the connector once. And as long as you pull as much data and you stay as close as possible from the source schema, anyone down the line can decide how they actually want the data to – And transform the data so it looks like what they want.

So there might be very, very specific use case, but today I think this use case should be addressed on the transformation side which comes after the extract and the load. So I don't believe – I mean there might be some API or some sources that require very custom work. I hope that you can get them to work through configuration. If not, people can always fork the connector, modify it and create their own version of it.

[00:25:44] JM: Let's say I have several sources, like a Stripe API, a Postgres database. I've got some other data sources. I want to periodically load all this data into a data lake. What does Airbyte do under the hood to sync all of those data sources?

[00:26:01] MT: So what we do is we sync data with what we call micro batches. So today we don't support any real-time use case that will come down the line, but we're really focusing on micro batches at the moment. So if you want to do that, either through the APIs or the UI, you just build, create a source for each of these different data silo that you have and you can decide if you want them to work in incremental mode or full refresh. Full refresh means every time it's going to run, it's going to sync the whole data and replace what existed before. Or incremental, meaning it's going to detect what has changed since the last run and only pull this data. So in the case of a data base, today we support like key-based replication. We want to move into a CDC and understanding like binary logs to give you like more performance. And for Stripe, Stripe out of the box in the RPI is super incremental. And then once it's done you just configure your destination. Today we support data warehouse. We can say that they are kind of a data lake today, but we don't support like data lake the way you would like a lot of people would think about, which is files on disk, on HDFS, on like delta lake. That will come in the next version of Airbyte. And then you let the system do it. And every five minutes, every hour, every day, it's going to look at the source, pull the data, transfer it to the destination and write it. And if it's something wrong, you will be notified about it and you can dig into the log or you can ask the community for any kind of support to solve your issues.

And, yeah, the transfer of data between source and destination happens in a pipe. So, today, because we don't have like massive, massive scale, we can get by connecting the source and the destination together directly. But we foresee that in the future we might have some queue in the middle to accommodate the volume.

[00:28:09] JM: What about incremental updates? How do you handle incremental updates?

[00:28:13] MT: So that will depend on the source. You have sources where it's impossible. So basically an incremental at the root of its definition is you take the snapshot you had yesterday, you take the snapshot you have today, you compare. You look what's different. This is your incremental. So you will have sources who have this feature embedded. Like the Stripe API is a good example. A lot of API actually pretty good at it where you can filter by and update that or credit that. So this is the kind of data that we will leverage. For a database it can be a field at the record level, whether it's updated that column or credit that column, we can detect that and just only like remember what was the last date we use and only take what is after that particular date.

And for the sources where it is not possible to do incremental, at that point we just do the full refresh and let the responsibility to the user to actually decide how they want to leverage this new full refresh. And that's actually one of the complexity of building this connector, is all these sources have different guarantees, different properties, and you need to make sure that your connectors know how to deal with it.

[00:29:36] JM: How do you handle normalization? For example, if I'm extracting data from an API like Stripe and I'm trying to get that into a SQL database, how does Airbyte load it in a correct format?

[00:29:48] MT: Yeah. So we've taken an approach that is like a lot of steps in how the data is being processed, and today what we do is whenever we read data from a source, we're going to encode it in some kind of a JSON object and we have some internal tables that we create on the destination side where we are going to load these blobs of JSON. And once this data has been loaded, we actually apply what we call normalization, which is taking these JSON blobs and actually splitting them up into different tables so that if the data was coming from a database, you get the exact same tables down the line. Or if it's an API schema, like making sure that you have one table for each of the record and each of the type of object that were in the source.

And today we use DBT under the hood to do it, which works extremely well with all these databases. So we leverage like the input schema, the blob data, and then we explore them. And this actually allows us down the line to – And that's something we're going to do. We already have a tutorial right now. It's very manual how to do it, but we can actually expose this resulting normalization schema and you can start using them into your DBT pipeline if you want to. And we want to, yeah, just expose that more and more, integrate better with like the analytics type.

[00:31:26] JM: So given that you're open source, you can be self-hosted. Can you explain why this is important?

[00:31:35] MT: Yeah. So for some SaaS API it's not a big deal because they are already hosted somewhere on the cloud, but not every company is okay with that, first of all. They don't want a third-party to have access to your data. So they will want to do all this data movement internally. Now there are other use case where it is out of the question to actually leverage a third-party vendor. For example, when you want to start replicating your internal user database or everything that is internal, you will rarely use a third-party to do it because it's your call data. You don't want that to be at the – You have responsibility first for the data. If it's customer data, you don't want to have the risk of having a vendor to just leak that information.

You also have the problem of scale. Sometimes these internal data sources or internal applications are high scale. So if you were to send that data to a third-party vendor on their cloud then it's a lot less efficient than if you were doing it locally. Anything you would add, John?

[00:32:43] JL: Not really, but I think it explains also why we see that if a company was using Fivetran right now. So we use them for 5, 10 connectors, but with us they will use a lot more connectors. And overall I think by commoditizing the data integration like this we will increase the number of connectors used within companies because we are addressing those use cases that couldn't be addressed before.

[00:33:11] MT: Yeah. And it also allows us to – Like the self-hosted piece allows us to address industries that are extremely regulated like healthcare, like fintech, finance where they have to do it internally for legal reasons.

[00:33:30] JM: Tell me a little bit more about how you handle orchestration of a running Airbyte instance.

[00:33:36] MT: Yeah. So today we have our own internal orchestrator. We're about to release an integration. Actually you can already do it, it's a bit manual right now, but you can start integrate within like Airflow, Dagster or any kind of DAG manager. So this is more like for the sequencing of all this operation. And now for like the process execution, we're also working with Kubernetes. We have an alpha version for like properly orchestrating these jobs on the cluster. But yeah, we still want to make sure Airbyte stays as a unit because there are things that we do to the data to make sure it goes – Like renaming sometimes like tables or renaming fields or actually redacting data that should never be persisted for privacy reasons or security reasons like a PII. This has to happen before it hits the destination. So at that point we need to have our own local internal orchestrator, but that remains extremely simple. But the goal is integrate with the existing orchestrator.

[00:34:48] JM: What's your monetization and business strategy?

[00:34:53] JL: So as mentioned before we are focusing on the open source part. So we're not focusing on the business model right now until ‘22. At that point we have two business models, one, which is the open core. This model where we will monetize hosting and management, but also enterprise features such as data quality, privacy compliance of GDPR, CCPA, or SSO, user access management. So that's one business model. That was the one we had in mind at the beginning, but the second one was brought to us by the community itself, which is what we call powered by Airbyte where we empower your YouTube offer integrations on your own platform using us. So us, like under the hood, with our API. And so you don't have to build those connectors anymore. You don't have to spend that much in terms of data engineering with those connectors, just the UI. You do your UI and use us in the backend.

[00:35:58] MT: And, interestingly, even though if today we are not focusing so much on monetization, the monetization deals that we have today are on the second use case, which is very interesting or course. Yeah, we have a few customers that we're actually onboarding using the powered by Airbyte product.

[00:36:22] JM: You're still pretty early in the game. What have you learned as you've been founding Airbyte and building the company? What are the biggest learnings?

[00:36:32] MT: On one side like the powered by Airbyte is definitely not something we've thought about. So it's like how the product got – We got informed on what direction the product can take from our community, which is actually a nice property of an open source project is you have access to a lot of people that are actually building and that come to you with their use case and they try to see how Airbyte could fix it. So this is for like the product side. We have also like another type of connectors that initially were really thinking about customer data, et cetera, but now we're starting to see a lot more around like security data, audit logs and people that needs to centralize this data for compliance reasons. So this is something we were not thinking about at first, so more like on the product side. John, anything you want to add?

[00:37:26] JL: Yes, indeed. So you always have these known unknowns, and one of those were to lock-in from other vendors, because data integration is a known problem. So a lot of companies have been doing this for years. So they have solutions in-house or they've already purchased. So what was the lock-in for them? And it was a big incentive for us, but we learned that there was no lock-ins. Only lock-in was really the yearly contracts, and that was a really, really good use for us. So we could really implement our strategy of land and expand. So land with a few connectors and then grow the usage within the company.

[00:38:11] JM: What are the biggest technical challenges that you're facing today?

[00:38:17] MT: I would say like every open source project, since you don't have so much control on like where your application is going to run and what are all these intricacies of implementation on your user infrastructure, you need to know how to get Airbyte to work in that context. So the difference in our environment in all is customers' cloud. And that's why like Docker is extremely, extremely – Like containerization is extremely important for us. And the other one is we're starting to get more people asking us for scale in real-time. So we need to accommodate that at the same time as we are continuing to grow the core to grow our connectors.

The other one I would say, maybe the last one, is how do we guarantee the reliability of our connectors. So making sure that we always invest in the quality and that we always invest in like the framework that we have to test these connectors and ensure that they work at scale and encoding like these thousands paper cuts that people encounter with connectors.

[00:39:28] JM: Give me your 10,000-foot view of the data engineering ecosystem. There're a lot of newer companies that are solving a lot of problems. You mentioned DBT. There's a company around that. You mentioned Dagster, there's a company around that. What do you see as the biggest problems in the data infrastructure landscape and the biggest opportunities for new companies?

[00:39:50] MT: It is actually very interesting especially when we talk with warehouse vendor, is we have a very solid data processing ecosystem. Like warehouse are just amazing today. They work. They work extremely well. They work extremely fast. They can accommodate crazy scale. And everything downstream has matured a lot as well. Like DBT is a very, very good example of it where suddenly they were able to put a model for how people can and encoding all the good practices for extracting insight from data and transforming the data. And after that you have also like all the visualization. You have like presets, you have mode, your Metabase, many other tools that are extremely good at presenting the data and presenting insights.

Interestingly, everything that happens before the warehouse we believe is really unsolved. And it's not just extract and load, it's also everything around data quality, data discoverability. Like once the data has hit your warehouse, like how do you discover it? How do you enable all these new roles in companies to discover the data and to understand what they can do with it? Like quality is obviously important. And in the recent years everything related to privacy has become a huge nightmare in the data world. Understanding that this piece of customer data has been derived 10 different times to arrive to an insight, how do you accommodate then like subject access requests or decision requests from your users with like CCP and GDPR? This has become a huge nightmare. Like the traceability of the data is not a solved problem and that's a huge pain point for a lot of companies, and they are proving like large enterprises such as risk of a legal issue that they are putting tons and tons of money to try to solve that problem. But ultimately it starts with where the data gets ingested. Like how do you at the root start to track and monitor every single step of where the data is going?

[00:42:08] JL: And it depends on your data infrastructure if you're centralizing everything or you're building a data mesh, but it's really the same. The difference will be like between data cataloging and discovery, but otherwise all the challenges that Michel mentioned around data integration, data traceability, linear lineage, quality, they are all similar to the same.

[00:42:35] JM: Does the data engineering stack feel like a mess to you guys or does it feel like developers who are data engineers are fully empowered today?

[00:42:48] MT: I think it's getting a lot better than what it was a few years ago. And I believe like warehouse have a large play into this improvement, where suddenly we can use like a standardized way of processing data that has existed for the past three years, which is called SQL, which is amazing for like not just – And warehouse are enabling, not just the analytics use case for leveraging data, but also the operational side, which is not just what can I extract as an information, but like actually how can I process that data to make it part of my product?

And at that point – And that's why also Airflow, Dagster have been so popular, is that suddenly they start exposing to many places in the organization how the data, what flows the data is following? What are all the transformations that are happening? And not everything can be centralized all the time, and this is helping a lot to have the sequencer for data processing.

I would say it's always going to be a mess one way or another maybe less than 10 years ago, but definitely over the past five years the eco system has completely changed. And I think we're lucky to be in that era today than when we were 10 years ago.

[00:44:12] JL: You have these open source – Emerging open source tools that are becoming standard, that DBT is becoming a standard. If everybody rallies around the study, it will make life easier because everybody will just integrate better with them. And you see great open source projects such as Great Expectations for data quality, you have Airbyte for editing creation. So I think like the ecosystem is structuring itself around standards. And yeah, we're very fortunate to be in there.

[00:44:46] MT: And one thing that is actually interesting is we used to have like this very monolithic software even in open source where they try to do everything from A to Z. And I think right now we're verticalizing how the data world is working. And this is becoming very easy with all the DAG manager, because suddenly everything is just a step. And at that point what you can do is for every single step you have the choice to take the best in this category, the best in class for this particular step or category. And yeah, and that's why like DBT is not focusing on extract. It's not extra focusing on load. It's just focusing on transformation. Great Expectation is just focusing on data quality and a little bit on data discoverability. And Airbyte is just focusing on extracting load. And we want to be the best in class in each of these steps.

[00:45:44] JM: All right, guys. Well, it's been a great conversation about data infrastructure and Airbyte specifically. Anything else you want to add?

[00:45:53] MT: Oh! Thank you so much for having us. It was great to be able to discuss it with you.

[END]