EPISODE 1182

[INTRODUCTION]

[00:00:00] JM: Data lakes and data warehouses store high volumes of multi-dimensional data. Data sources for these pieces of infrastructure can become unreliable for a variety of reasons. When data sources break, it can cause downstream problems. One company working to solve the problem of data reliability is Monte Carlo data. Barr Moses and Lior Gavish are founders of Monte Carlo and they join the show to talk about data reliability and the overall landscape of data infrastructure.

[INTERVIEW]

[00:00:35] JM: Barr and Lior, welcome to the show.

[00:00:37] LG: Hi, Jeff.

[00:00:38] JM: So you both work on Monte Carlo data, which focuses on data reliability. Could you explain what problems lead to inconsistencies in data?

[00:00:47] LG: So there's a bunch of different issues that can result in bad data or inconsistent data, and to get to that you kind of need to think about what are some of the different factors that go into that. The first factor that we're seeing is that companies are starting to use more and more data sources to feed their data analyses. And the more data sources you have, the more things can break and the more external factors impact the quality of your data. The second factor that impacts bad data is that people are doing increasingly complex transformations and analyses with their data. And so there're many more steps on the way from source data to data product like the BI report or machine learning model. And every single step adds complexity, adds things that can break and logic that can be wrong. And so the more you have, the more likely it is for something to break on the way.

And then the last factor is that there are many more people that are involved in building data pipelines, right? So the more people are involved, the less anyone has kind of a central view or a consistent view of everything that's happening to the data. And so every single change that comes from a different person might impact the work of someone else. And so that can lead to breakages and errors and issues with data. And so all these things are kind of coming together to create various problems. And in a way it's analogous to software, right? Every single line of code or every type of input that we might get from users can create new conditions and new things that we haven't thought about before. And so things break, right? And the same thing is happening to data flowing through pipelines.

[00:02:32] BM: Very specific example of that could be, let's say, there's an engineer or someone upstream that makes a change to your website and that has unintended consequences on data that your marketing team is using downstream. If there's no communication about that change, the marketing team might be using that data for their products in a way that has negative implications on the business or is leading to sort of bad decisions based on that data.

[00:03:00] JM: These are all understandable problems, but it seems like these are insurmountable for some kind of software tool or platform to solve, because it's kind of like human error. Like you can't have some single tool that prevents everybody from creating bugs.

[00:03:16] BM: It's a really good point and it's a question that we sort of also had starting to think about this. And one of the things that we like to think about is similarly to application downtime. Apps can break for millions of different reasons, right? You can't really avoid or prevent a bug from happening. However, there's a certain kind of set of metrics or framework that we use in order to create visibility into the health of those systems to, at the very least, know when things break. And so that's kind of the whole methodology of observability that's emerged in software engineering in the last couple of decades that allows engineers to know when things break.

And so any engineering team has a solution like New Relic or AppDynamics that helps them understand their system in a way that creates sort of an understanding of the health of their applications. And so their applications can break for many different reasons, but there's still a coherent set of metrics that you track, like performance, latency, CPU. And so when you think about the corollary to data, again, you can say, “Okay, data can break for millions of different reasons. Human error is really hard to avoid. There can be very many bugs.” However, we actually talk to hundreds of data teams at this point to really sort of ask them like what are the common reasons for why data breaks. Tell us about sort of an incident that happened to you and then also tell us about how you actually detected that. Like what were the signs that told you that something broke and why?

And so using that, we actually came up with a framework or a consistent set of metrics that we use and what we call data observability that helps us understand the health of your data in a very similar way to how New Relic helps engineers.

[00:05:00] JM: So you found the problems. What did you have in mind for solution?

[00:05:03] LG: I think solving the problem fully automatically is of course very challenging. And, again, taking the analogy from software engineering, when you create observability, your goal is, A, to let people know that things break before their customers do, right? Like that's the main function of an observability system. And then the second piece is once you know there's an issue, how do you get to the root cause as quickly as you can, right? The solution, there could be a lot of different solutions because the problems are varied. It might be a code fix or it might be a new code that handles a new situation or it might be restarting a certain ETL system that failed. So the solutions are varied, but our goal with observability is to really help teams know that they need to solve these issues in the first place. And today that's a big challenge, because for data teams, they usually – Or a lot of them learn about problems from their customers from the people that use the reports they're generating or from people that are using the data to train machine learning models. Or if they're feeding their data into digital products, it might come from the company's customers. And so the challenge is really to reduce the time to detection to learn about these problems earlier and then to have enough

visibility into the system, enough metrics to be able to say, “Okay, the problem that we're seeing is actually coming from this particular source and because of this particular change. And then we learned that the data engineers and data analysts are pretty good at solving these issues once they know they exist and once they can pinpoint where the issue happened.

We definitely have some thoughts about how we might automate some of those solutions and help drive automation through that. But to be honest, we started out from just highlighting that issues are happening and helping tracking down the root cause. And that's been a tremendous help for a lot of data teams that struggle with that.

[00:07:08] JM: What's the best integration point for a data reliability tool?

[00:07:11] BM: So the interesting thing about data reliability or kind of the inverse of that is data downtime. So data downtime sort of refers to periods when your data is wrong, you’re inaccurate or otherwise broken for some reason or any of the reasons that we talked about. The interesting thing is that I think in the past you probably had sort of the main focus of making sure your data was right was upon ingestion. So you needed to make sure that sort of the garbage in, garbage out, right? You need to make sure that whatever data you're ingesting, that it's clean and consistent so that you can use it downstream with confidence.

However, in sort of as sort of the data stack and architecture has evolved in the last five to ten years, data can break really anywhere, not just upon ingestion. And so it can break in the data warehouse. It can break in the transformation, in the ETL or even some logic in your BI or in your ML. And so to Lior’s point, to truly allow data organizations to be the first to know about data breaking, you need to be able to provide coverage that goes beyond ingestion and really sort of connects to as many sort of data points as you can. I mean, so our goal is to provide this end-to-end coverage for data organizations including data lakes, data warehouse, BI and ML.

[00:08:30] LG: And that's a really important point. We believe that – Or a big part of the challenge with data issues is that there's a long chain of things that happen, as I mentioned

earlier. And if you really want to help teams deal with that and kind of accelerate the resolution of these issues, you really need to get that end-to-end coverage from the source all the way through the leg, the warehouse, and the kind of end product. And so end-to-end connectivity is actually critical to get the job done here.

[00:09:03] JM: Could you give me an end-to-end use case about how data reliability tooling could be used to prevent errors?

[00:09:09] LG: Totally. So let's take a concrete example, right? We can take an e-commerce example here, one of our e-commerce customers. So they essentially have a lot of events coming from their website and from their mobile application. These events using an ETL land into the company's data lake. From the data lake the events are aggregated and stored in a warehouse where it gets used by analysts to create reports that the marketing team mostly uses in order to make decisions about different campaigns and about where the company might invest its acquisition efforts going forward and also how to improve the product and the conversion rates in the company's e-commerce platform.

And so that chain of transformations can introduce a lot of problems. And I'll give a particular one. The engineering team with that customer actually made a change to the mobile application. That change had some unintended consequences. One of the fields and the tracking events that were critical for the company actually got removed and wasn't sent any longer, which broke some of the transformations along the way and led to incorrect numbers that the marketing team was using to analyze the performance of marketing efforts and of the application. And that change is very subtle. It's hard to find and usually would have taken a long time to detect that and then to find out what happened. But if you have this end-to-end observability, and in this case you can actually detect the problem as soon as the data lands in the data lake, and as soon as you detected that issue you can also trace it throughout the pipeline and all the way to the warehouse and to the reports that were impacted. You can actually tell, “Hey, this field that used to be here, it's now gone.” And the fact that it's gone might impact this and those reports that are being used by this person on the marketing team, right?

So the data team can really focus its attention and understand that, A, a change happened. B, what was the change? And C, who's going to be impacted and is this something that requires attention? And so this end-to-end observability really gives you the full picture of what's going on? What's important? What needs attention and how to solve it? And in this case solving it means going back to the engineering team and asking to fix the bug, but that's a result of knowing a lot of pieces of informations about how data goes all the way from the mobile application to marketing activities, if that makes sense.

[00:11:58] JM: Could you tell me more about why you started Monte Carlo data?

[00:12:02] BM: Sure. So actually this was based on kind of our prior experiences working with the best data organizations and both experiencing this personally, but also seeing how pervasive data downtime was as a problem. So prior to Monte Carlo, I was a VP at a company called Gainsight where I led our team that was responsible for our customer data and also worked with our customers on their sort of data initiatives. And one of the things that really struck me was that as organizations were trying to become data-driven, we're trying to use data for real-time decision-making to feed their sort of digital products. The thing that sort of came up the most or kind of kept data organizations up at night was really sort of the ability to know, “Can I actually trust this data?” And oftentimes in order to answer that question they needed to answer some really basic fundamental questions about the data like when was it last updated, and where does this data come from, and who else is using this data? And it was incredibly hard to answer those questions.

In fact, most data teams did not know the answer to those questions and didn't even know how to start answering those. And so in my personal experience actually running sort of a data team that was responsible for this, the way that I experienced this was we had sort of a set of kind of reports or dashboards that we were using for executive decision making based on data that we were collecting in the product. And on a regular basis, that data was wrong. On a regular basis, it broke. And so I would wake up on Monday morning to find these series of kind of issues that hit our reports, and it was really hard for me to know whether it was because

there was a change made in a website or was it a job that failed somewhere or another kind of like silent failure or maybe there was just some logic that changed without me knowing.

And so, for me personally, it was incredibly hard to sort of learn about these issues. We were often times the last to know about it. It was the consumers of the reports and the data that let us know about it. And also it was really hard for us to pinpoint the issue anywhere along sort of the chain or the pipeline where exactly data broke. And then when we did detect it, it could take us weeks to resolve it. And, overall, this entire process was very, very manual and taxing on my team. And it sort of blew my mind. I was like, “How is it that we have all this some sophisticated architecture and sophisticated systems to store and collect and transform data but it's actually incredibly hard to know when it breaks or when it changes and incredibly hard to know the downstream implications of that?”

And so zooming out of my experiences, I started talking to other folks and realized that this is sort of a widespread problem. And inspired by a desire to make this easier for data organizations, teamed up with Lior. Had experience with this from a different perspective and decided to build a solution that will give data teams the same confidence that engineering teams have as they work with their architecture.

[00:15:04] LG: Yeah. For me, kind of similar story but a different perspective, if you will. I was running the engineering team at a company called Barracuda. It's a cybersecurity company and I spent most of my time on building kind of machine learning-based systems for fraud detection. So the way these things generally work is you pull a lot of data about your customers environment and a lot of data from third-party sources about threats and you kind of merge these datasets together to help detect security threats and fraud inside your customers environment. And we use machine learning models for that. So we would train, we'd take the data and train machine learning models that automatically detected security threats in real time. We were a software engineering team, right? Not an analytics team. And we're pretty good at managing the reliability of our system, right? Like there's a lot of methodology around DevOps and all that good stuff. And so we became pretty good at managing the reliability of

our infrastructure and of our application and it was pretty rare for an application issue to impact our customers.

However, when we had data issues, when we weren't getting the right data or when the data was wrong or when we pushed a bad transformation, we would actually cause issues for our customers, right? We would get to the point where we're disappointing them and providing bad service. And that really struck me. Like how come there's a lot of methodology and tooling around controlling the application and the infrastructure, but there's no good way to understand healthy data that's flowing through the system is. And the more products are built today around data, around machine learning models, the more you need that visibility and you need to manage the reliability of your data as much as you need to manage application and infrastructure reliability. So that kind of really struck me and made me think a lot about this problem and kind of joined forces with Barr to help data teams deal with that.

[00:17:08] JM: Can you tell me about how data observability changes at different phases in the data pipeline? So like how data observability in the data warehouse compares to data observability in the data lake versus the ingest point and so on?

[00:17:21] LG: That's a great question. I'll separate the answer into two. So the first part is what is data observability in the first place, right? And that's something that we needed to define, right? Like application reliability is well understood by now and there's a good set of metrics that everyone knows they need to follow and there's increasingly better products that help you instrument your system and extract those metrics and get value out of them, right? Monitor the health of your system and resolve issues.

We needed to define what those metrics are for data, right? And so there're kind of five pillars if you will of data observability. So the kind of first thing that we believe you need to measure about any data system is data freshness, which is the question of like how recent is the data that I have. Is it up to date? Has it been updated? The second pillar that we believe needs to be tracked is data volume. So track, did I get the amount of data that I'm expecting or am I getting a lot less or a lot more? Both of those are indicators of issues. The third is schema. So

kind of asking ourselves like what fields do I see in a data? What types are they coming as I expect them to? Are there things missing or are there things that have been added that I need to know about? Are there things where the data type changed and I need to adjust my code to that?

The fourth piece is what we call distribution. So this relates to the actual data points. Am I getting data that's within the expected ranges? Am I getting too many nulls? Am I getting duplicate values or not? Things like that. And the fifth piece is data lineage. So the idea of understanding the dependencies between different data assets whether they're on the lake or the warehouse and how various tables are derived from others and what depends on what and what are the downstream implications, right?

So to answer your question, Jeff, these five things are actually consistent end-to-end, right? Like we want to measure those same things around your data lake, around your data warehouse and through the BI reports that you have on top of that. And that's what we built for. Where things get really, really different is the actual implementation, and that's actually a lot of where the challenge is from our perspective. How do you measure those things across your warehouse and lake? And it's a very different challenge. Just to give a taste of it, warehouses tend to be kind of integrated systems where you can access metadata and logs and statistics all from a single API and from a single set of credentials, whereas data lakes are kind of mix and match systems where you have different types of systems for metadata and compute and storage. Like you might have a Hive cluster to manage your metadata and an S3 bucket to store your data and then various query engines on top of that, like Presto or Athena or others. And the challenge with data lakes is to actually integrate it with all these different components and different systems and extract those observability metrics that I described earlier.

And so the implementation varies, but the end product is the same, right? And that's what we're trying to create in our solution as a consistent experience across lakes and warehouses and other tools. The underlying implementation of course is very different and much more challenging with lakes.

[00:21:01] JM: Is there an analog to data observability tooling? Like do you compare it to a logging tool or a monitoring tool or is this a different class of product?

[00:21:12] LG: We believe it's a different class of product. It's a new category of solutions that are emerging. It has a different end user. So for us it's data engineers and data analysts versus DevOps or software engineering teams. The instrumentation is very different. So whereas kind of a traditional observability tool will connect to your containers or servers or serverless functions and databases, we focus on things like warehouses and lakes and BI tools. So it's a very different instrumentation process. And also the kind of experience or information that the tool provides is very different, right? All the observability pillars that I mentioned earlier are very different in practice from what you'll find in kind of a New Relic or a Datadog.

Having said that, I think there's a ton to learn from how observability has been done with applications. There're a ton of analogies that we can take from the world of New Relic and datadog and apply to the data world. So there's a lot that's similar, but there's also a lot that's different. And so while we learn a lot from DevOps, we actually consider a different category of tools that's geared towards a different audience and use cases that are similar but have their own intricacies and quirks.

[00:22:34] BM: One of the interesting things that we're seeing is that, with our customers, once they have this information in their hand, information around data observability metrics like Lior described is that sort of the natural next step that they take is sort of akin to what they see in the observability space. So whether that's starting to track SLAs for their data, having dashboards that actually monitor data downtime, trying to get to sort of five nines of data availability. So there's definitely a lot that we can lean on and I think it's good news. We don't have to reinvent the wheel. But it's certainly a different kind of product with different kind of users.

[00:23:10] JM: Tell me about some of the hard engineering problems that you had to solve when you were building Monte Carlo.

[00:23:14] LG: There's many, but one thing that stands out is it's exactly that, right? What we've been talking about. Like we want to provide a consistent experience and measure all these observability metrics on various types of source systems, right? Data lakes, data warehouses, BI tools, ETLs. And so one of the biggest challenges of has been how do you build our system in a way that allows us to extract information from a lot of different source systems, and we have probably a dozen integrations today and more and more that we're building. So how do you integrate with very different systems but then take all of that data and align it, normalize it in a way that can be then consistently processed to identify data issues and help with resolution, right? So there's really a lot of complexity to that and to really mapping dependencies across different systems in a consistent way. So we've worked a lot on our data models and our pipelines to be able to do that.

And what adds kind of a degree of complexity here is that – And this is a really kind of important thing that we had to tackle from day one, is security and compliance, right? So since we are connecting to our customers holiest data, the most secret and the most sensitive data that they have, we are scrutinized from a security perspective and we need to be really the best in terms of guaranteeing our customers' data security and their compliance with various regulations including GDPR and PCA and a bunch of others. And so we need to build our system in a way that allows our customers to run a lot of our application on their side and still enjoy kind of an experience that's managed by us that's served by us. And so we have this hybrid architecture where when you use our product it feels like a SaaS product. You can go to our website and sign in. But in the background, we've actually installed big parts of it on our customers' infrastructure. And how do you do all this observability magic while keeping the sensitive stuff on the customer's side and only exposing kind of metadata and statistics to the end user and to our cloud environment? So that definitely took a lot of efforts to get that right and to get the security model around that tightened up so that infosec team feels comfortable with it and let data teams use it for their purposes.

[00:26:03] JM: The data engineering landscape is moving really quickly. There're a lot of different pieces. How do you keep up with all the different integrations?

[00:26:11] LG: That is a great question. To be honest, we're mostly aligned with our customers. So we work very, very closely with the companies that have adopted our solution and kind of concentrate on those integrations that those companies require. A big tailwind for us has been the fact that there is a huge transition to the cloud into a more consistent set of data systems. So the fact that people are using more and more of Snowflake and BigQuery and Redshift on the warehouse side, and Tableau and Looker on the BI side, and Databricks on the data lake side. So those cloud technologies created more and more consolidation in the market and allowed us to actually give a lot of data teams a good amount of coverage without having to build literally hundreds of integrations.

So we're really focused on those technologies that are emerging and kind of widespread and we work pretty closely with our customers as they adopt new solutions or add more parts of their infrastructure. We build with them and we make sure that they have the end-to-end observability that they need.

[00:27:24] JM: How big does the data engineering stack for a particular company need to be to where they start to need to think about something like data reliability?

[00:27:36] BM: I think there's a number of kind of different stages that a company goes through, and part of it is you'll see in the evolution of the stack and part of it is kind of in actually the number and kind of people that are on the data team. So in the very early days, if a company is just getting started, there might be sort of one kind of person that does all things data. Sort of a data analyst data engineer, I mean, one. Often times you might actually just be using Excel or something like that to run reports or to try to understand, let's say, cohort segmentation of your customer base. So answer pretty basic questions about your business. In those days, in the early days when they're sort of one to three to people in the data organization, oftentimes if data breaks or if data changes, it's fairly easy to know and track because there aren't very many systems and there aren't very many dependencies.

I'll take the other extreme of that spectrum where you see companies like Uber and Netflix where there're thousands of data consumers relying on data in real time every minute of the day. And there're probably thousands of data producers who are making changes, adding new sources, third-party data that changes. And so in that kind of very, very sophisticated complex environment, obviously, reliability is kind of an everyday reality. The question is where in the middle is that do you need to start thinking about this?

And so we see folks who have been in the data space for a while have experience with this kind of at scale know that this is something that they need to start thinking about from day one. So as soon as you have your basic data warehouse or data lake and your very first BI solution, that's already the time to start thinking about how do I make sure that the data that's being used and the data that's being processed is actually accurate and can be trusted? So I think it's the kind of thing that you can you can't start too early. But as soon as you sort of hit sort of the issues of I'm not sure where this data is coming from. I'm not sure who made the change. You need to start sort of thinking about a data reliability solution.

One thing that's exacerbated this, I would say, is sort of the move to remote and the move to work from home where you can no longer sort of shout to the other person the other end of the room and say, “Hey, did you happen to make a change here or do you know which report can I actually use or what datasets should I be looking at?” And because we can no longer do that, I would say that the threshold for that has moved to even sort of kind of lower. But I would say you know starting from having five or six or so data engineers or data analysts is kind of when you want to start thinking about making sure the reliability of your data and that you have observability for it.

[00:30:27] LG: My litmus test that I've seen work is at what point you get to a place where there's no longer a single person that knows every part of the system and every single table? And I think heuristically it happens at around five or six people on the team. But once you get to that point, that's where you start need to being more methodical and more systematic about how you manage your data reliability in your data pipelines and that's where data observability becomes really important.

[00:30:56] JM: Is there a particular engineer at a company that will be responsible for data reliability or is it just everybody who's interested in the data engineering stack at all subscribes to a channel in Slack and then they receive updates about it?

[00:31:11] BM: I think what we're seeing is closer to the latter where sort of every person that is either a data engineer or a data consumer needs to be a responsible user of data. And so whether it's subscribing to alerts on Slack to know about changes, that's one use case. The second use case is actually tracking sort of the monitoring of it proactively. So what we're seeing is that once you have this information, before you actually make a change, whether you're adding a new data source or changing a particular pipeline or really any change that you want to introduce in your system, you can actually go proactively into your data observability solution. Understand the implications of this.

So, for example, if I'm changing something now, what are the data assets that depend on this? If I need to change a field type, what are all the data tables and all the reports that rely on this particular field? And then who needs to know about this? Like if I make this change, do I need to introduce any sort of fixes downstream or do I need to notify the stakeholders? And so even before making those changes, you can do that in a more responsible way in which results in less data downtime overall.

So I would say data engineers subscribe both sort of to alerts and also proactively use the system to know about changes. And on the data consumer, having all this information means that you no longer need to sort of go bug people and ask like, “Hey, should I be using this report or this dataset? Or where did this come from? Or what are all the upstream dependencies of this? Or is this data coming in as expected? Instead of kind of having to sort of chase around the organization to understand that and/or just make a bet on some dataset and hope that it's accurate, you can actually know this information to begin with and actually move faster and focus on kind of revenue-generating activities. And so in a way, one of our goals is to actually democratize data reliability. So make it something that anyone in the

organization can have at their fingertips so that they can use that when they make decisions and when they're sort of building data systems.

[00:33:25] JM: Any predictions for the next five to ten years about how the data engineering stack is going to change?

[00:33:30] LG: Well, clearly we're going to have a lot more tools. There're so many startups building cool things for data engineering. So I'm really excited about that as we're kind of automating and standardizing things. Obviously, from our perspective, observability going to be really huge as people build larger and more complex systems. They're going to need to figure out how to better understand those as we've discussed today. The other thing I think is happening in data engineering is the idea of data platforms. So, increasingly, we're seeing it with our customers. It's not just one data team that is providing data products to the entire organization, but actually a collection of teams where every single function or business unit or product group has its own data team. And there's a central kind of team that manages a data platform and offers services to those more specialized data teams. And I think there's going to be a lot of tooling around that, around how do you not only build the data pipeline, but how do you build it in a way that it can serve other people that are not on your team and can allow them to build on top of that? And the concept of data meshes might be helpful here or there are other ways too, but data platforms are here definitely something that we're seeing more and more.

And then a kind of final piece is people are building hundreds and sometimes thousands of dashboards to track various aspects of their business and various data assets that is incredibly hard and perhaps not sustainable at scale. And so I think there's going to be more automation and more AI to help companies automatically analyze data and automatically point people's attention in the right direction and help automate that process of analyzing data and tracking it. So these are some predictions for the data space.

[00:35:36] JM: That seems like a good place to close off. Do you guys have anything else to add?

[00:35:38] LG: No. Great chatting with you, Jeff, and I've been a long-time fan of the show. Spent a lot of time listening to your show. So excited to be here today and thank you for having me.

[END]