EPISODE 1452

[INTRODUCTION]

[00:00:00] JM: As companies move to a Spark and Lakehouse architecture, they're realizing that the data tools are lagging behind. You often need to be a programmer to effectively use Spark and Airflow. There are some low code ETL tools, but is that enough? Companies want to treat their data pipelines like mission critical apps. They want DevOps for data with Git, test coverage, and CI/CD. In addition, companies need end to end visibility of their data pipelines, including monitoring, metadata, and column level lineage. Where are the data tools for these capabilities? And how do non-programmers use these?

In today's podcast, you'll see how a Prophecy's Low-Code Data Engineering Platform is built on Spark and Airflow and provides a complete solution for developing, deploying and monitoring data pipelines. We'll speak with Raj Bains who is the founder and CEO of Prophecy. Raj has spent two decades building powerful tools, Microsoft Visual Studio, founding engineer for CUDA at Nvidia, Product Manager for Hive at Hortonworks. He is passionate about making organizations productive with data and making the lives of data engineers better by reducing toil, and increasing their joy.

Also, if you like what you hear, Prophecy has a special offer for our listeners. You can go to prophecy.io/demo and put in SED for the referral code. If you decide to purchase within three months of the demo, you get 10% off. If you purchase after three months, you still get 5% off. Thanks for listening. And thanks to Prophecy for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:01:42] JM: Raj, welcome back to the show.

[00:01:44] RB: Thanks, super excited to be here.

[00:01:46] JM: I want to start by talking about the modern data stack. So we've reached a point of maturity, where a lot of organizations have picked either Snowflake or Spark as their main source of data processing. I guess the first question I have is, what the dichotomy is between companies that choose Snowflake versus ones that use Spark?

[00:02:16] RB: I think I would start with saying that, the cloud stack is being built bottom up. So you had the AWS, the storage, the compute, then we started building the processing engines. You have Snowflake, and you've got Spark and Databricks. Now, we are at a point where both of those engines work pretty well and do their use cases pretty well. We are now seeing that the tools are getting built on top of that, and the customers are kind of self-selecting for what their priorities are. And similar things have existed on premise as well, where you had Spark like engine in Ab Initio and Teradata instead of Snowflake.

So what we've seen play out in the last generation, and we're going to see now is a lot of the smaller companies, or some of the large enterprises for smaller use cases, right? It's a department and they have a use case. They are going to say, I am going to optimize for simplicity, and when they do that, they will say, “Okay, I've got a little bit of data, I'll put it in Snowflake, and I will do analytics on top of it, my analytics is not very complex.” Okay, so they are going to choose the modern data stack. And the modern data stack has gotten built quickly, because it's simpler. And simpler use cases, that's great, right? They're out in the market quickly, solving those use cases, and people are getting value from them.

Now, in our customer base, the people that we talk to, are deeply engaged with our midsize to large companies. Now, these are all smaller companies that have large data. This is where they have high volumes of data, some complexity in pipelines. It's like end scale. It's one thing to put 20 pipelines into production, you can do it very quickly. When you're trying to put 10,000, then it's a whole different game. You need to have different kinds of processes. So what we are seeing is that these companies with larger data sets, who are cost conscious, who are looking at scale and complex use cases, they are more going towards Spark, they are more going towards Databricks. Now, they might come back and again buy Snowflake, but they're also – ML is a must have in these days for these big companies.

So you have the data, the data comes into the data lake, and now, I'm going to process it there and get it in a ready state. I can push it into Snowflake also. That's great for some BI, but I also want to do ML and then I'm going to use TensorFlow or whatever is the hottest thing in machine learning these days. And I'm going to use that so the larger companies are very much skewing toward what might be called the data lake, rather than just the data warehouse, and data lake might come with a data warehouse. So it's data lake and data warehouse.

[00:05:11] JM: Right. So would you classify it as by any partition that has to do with legacy versus non-legacy? Or it's more about, there are some companies that for whatever reason, want unified data storage and processing in the same place being Snowflake? And other companies want a separation of concerns there, and they're choosing Spark, because they have data, maybe in a variety of places?

[00:05:48] RB: Yeah, I would put two things. One is that a lot of the conversation in the Bay Area in the startup community is based around what smaller companies do. A bunch of startups and that's what the VC startup ecosystem, and their problems are not that complex, often, right? They're just starting out with data. So I would say that what splits them is the size of the problem they're trying to solve, and if you're trying to solve a much simpler problem, then you are skewing towards Snowflake only. If you're trying to solve like, for example, if a huge part of what you wanted to do was machine learning, then would you put all your data in Snowflake?

One part is like, they have a simple use case, let's say they don't want to do ML now. Snowflake might say, and you're not against it, Snowflake is great. Our customers use Snowflake as well, more for BI. But then, are you going to take that data out back to the data lake and use it for machine learning? Are you not going to do machine learning? So one big split is people are just saying that, "Hey, I have BI use case, and I have machine learning use case. If I have two use cases like that, I am not going to put all my data into the BI stack." And then take it out for machine learning as an afterthought, right?

The second thing is just lack of experience, I would say, which turns out to be very, very common. Because for the large enterprise, and you might say legacy enterprises, but actually legacy enterprises are using a much better stack than all the cloud people are, all these new cool startups, like their data engineering really sucks, and they think they know what they're doing. They haven't seen the last generation of tools or product. So, they don't know. Let's say I had experience of 20 years, and I paid through my nose to Oracle, and through my nose to Ab Initio, we have customers who are coming to us from Ab Initio who are paying 20 million a year in licensing, approximately. We don't know the exact numbers. But that's the estimates we've heard.

Those people are not going to go and get locked into a Snowflake now. Because they know that what happens when you get into scale into a proprietary solution, they are like, "We like having the insurance of having a Spark there." I think, it's the simpler use cases, it's simpler in terms of the data and complexity, simpler in terms of the business problems you're trying to solve, and lack of experience. I think those use cases are definitely skewing snowflake, more complex, more scale, more experience is skewing Databricks.

[00:08:22] JM: Got you. Now, the product that you're working on, as we discussed in the previous episode is Prophecy, and you're building a data engineering tool set that has a focus on Spark and Airflow, and also is UI driven. So some people might see this as kind of pulling off the tail winds of low code or no code, but it's not exactly no code. But it's UI driven, basically, solid UI over Airflow and Spark. Obviously, the primary vendor for Spark is Databricks, and if you're looking at the Databricks ecosystem, then you get a lot of things that you want on top of Spark. You get the notebook functionality, potentially, you get the data lake functionality. And I guess I'd like your perspective on the relationship between Databricks, and the Spark ecosystem, and how the impact of Databricks fits into what you're doing.

[00:09:35] RB: So typically, as markets mature, I would look at it as an entrepreneur trying to build a company and trying to place markets in my own head, you ended up with a – for every kind of technology, you have a premium product and the base product. It's like yes, I can buy a branded table or I can buy the Amazon Basics off table which will be a good enough table, it will do the job. So that way right? So Databricks is the premium Spark, and then every cloud provider provides you with a basic Spark. So that means that, at least to me, as a business, who cares a lot about lock in or I know that Databricks prices will always be reasonable or not beyond a certain point, because there's a very similar thing available. Now, that's number one.

Number two is going back, and so that's kind of the relationship between Databricks, Spark and that. Now, Databricks, Spark will also provide a lot of interesting functionality on top, right? One of the things I worked at Nvidia, I really liked how the organization. It was well run, they were very capable. That's a company and set of engineers I would bet on Databricks. So they are able to build a lot of pieces of the stack, and if a challenge comes, let's say Snowflake did, they are going to build something and I would bet on them being able to build whatever is needed. So I think that for us, as a company realized, that's the horse I'm going to ride on.

Now, the next thing is, but when you look at like – so let me give an example of our users. Now, the other thing is the third piece that you talked about, which was like, “What does Databricks provide versus what does Prophecy provide?” If I look at our users, let's say, one of the top financial firms, we have a few banks and a few credit card processors running on us. So they use the best of last generation technology. I mentioned this last time as well, they used Ab Initio. Now, one thing to realize, I think, maybe if I go back a little bit, like when we were, and maybe we can come back to this thread, when we were selling Hive and Hadoop to those people, they could easily see that the product was inferior, but it was cheaper. So it's not like somebody went and said, “Hey, you have Teradata, my Hive is better.” You said, “Hey, Teradata is expensive. I have this Hive, it's kind of sort of not as good, but it's cheaper. Save some money.” So that way, I go and look at and we've got lucky in that. We go and look at Ab Initio, and see what the people already have and what's working for them.

So now, this is a very interesting dynamics that happen. What happens is, you have a bunch of a good number of visual developers who have had their ETL working beautifully, for 10 years on Ab Initio. You ask them, “How is it working?” In Ab Initio, they have a distributed high performance processing engine, very similar to Spark, supports relational and general purpose distributed programming. On top of that, they have visual drag and drop, which gives you productivity. You can drop into code should you need to, but 90% of the time, you don't need to. You can create your own custom visual components. That's great. Now, everybody's using the same pipelines. What happens is, you build one data engineer builds a pipeline, they leave, the next one comes, somebody answers on kind of pager duty, and suddenly, they need to fix something. Everybody can understand each other's pipelines. So that is great.

Then you go to the data analysts, and they understand how every value was computed. They have end to end column level lineage, and there is a summary saying, “This is the business definition that they tag on to it, and this is how it was computed.” So now, when they're making business decisions, they know exactly how. Data analysts are able to – they have a business rules engine that they can take this data, apply some common rules that that it’s like the industry formula for this, the industry formula for that, and they can apply those.

The governance people are like, I have personally identifying information. It is in this column, I tag it, and it shows up on every data set with the restrictions on it. So they had that, they had basic performance and quality reports that came out of the box. This is running, and now these new people show up, they're like, “Hey, you guys are okay developers, we are awesome. We have Spark, let's move to Spark open source and save a bunch of money.” They move to Spark. And then, you go and ask their metadata architect, they're like, “Oh, we are running blind. We went 20 years behind in technology since these people came in.” You’re like, “What?” Like, “Yeah.” He's like, “I spent months and months sitting with these people trying to figure out my governance, that data analysts don't know how the value they're using was computed. Half their dashboards are wrong and it takes forever to get anything done." The visual developers are like, "It takes me 10x longer to build pipelines, and it takes everybody 10x longer to build pipelines, what the hell's going on?" And then the Hadoop people go back and say, "Oh, we need to get better engineers and more engineer." Initially, that's the narrative, right? Because that will solve it. But that still doesn't get solved your lineage problem, that doesn't solve your search problem. There is a lot to data engineering, and basically, if you think like the best barrier companies, have it, you'd say, "Go, Google does this." It's like, "No, this sucks." And you're like, "Why does Google suck at it?" It's like they're just throwing bodies for – and we can come back to the cost optimized, they're building cost optimizer. Nobody in Bay Area has their data engineering together.

The best instance, and like I said, I've talked to hundreds of companies, the best are the people who are like, "I went to airline in Texas, and they didn't touch Hadoop, and they were on ebony show, that they got 15 years ago." They have the best learning data engineering I have ever seen. And they are like, we will move to the cloud once we have the same stack.

So I'm just saying that – so coming back, so Databricks now, coming back to your question, it was a little bit of a tangent. But coming back to your direct question, where you're saying, "Okay, so what does Databricks provide and what do you provide?" It's not just a UI for building stuff. When we say local development, it's like, as you're doing visual drag and drop, we are going into a Spark cluster instrumenting it. You can see the data as it flows through. The first time you do iterative development, you create a new thing, you are getting autocomplete. It's like saying, "Hey, you are just a tool on top of a text editor." It's like, "Yeah, but you know how much productivity I get out of using IntelliJ?"

So that's number one. Then number two is like, "Okay, how are you going to schedule the jobs? How are you going to monitor the jobs? How are you going to explain to the business users what each value means?" Now, what happens if somebody who's a data scientist says, "I want to do some data engineering?" Do they have come into your system? Do you have something – what happens when some data analysts say, "I want to do some data engineering in line of business." They say, "Oh, no, no, you don't have access to it, send a ticket to the central data platform team, like it was 1980, '90?"

The basic thing is, the amount of tooling that needs to get built on top of processing engines is as big as the processing engine. So Databricks is half the solution. A very good version of the half. The other half is like, "Okay, you started using Databricks." You go ask a Databricks customer, what's your data quality story? How quickly are you getting data pipelines into production? Do you have linear? Do you have search? Can you use reusable? For our customers, what they do is like, you have a health insurance, and the data is very similar data that it will be coming from 20 different places. In our visual thing, they have a subgraph that cleans that data. They kind of change the fields of the name, and through the same sub graph with slightly different configuration on 20 different data sources. How are you going to do that?

Basically, what I'm saying is, Databricks is the data processing layer. On top of that, you need a full productivity platform, a metadata platform, like where is your business metadata? Is this called heart rate? Is this column what? Is this insurance? What does this value even represent? So is there business metadata in Databricks? No. Is the business metadata in any cloud? Does Google Cloud have it together in AWS? No.

[00:18:07] JM: There's like a whole wash of startups doing metadata, data management, right?

[00:18:12] RB: That is correct. But the problem is, the challenge is – I had one of the VCs say, “Hey, why are you building a metadata system on the five startup?” And my answer to him was, “Five startups getting funded by five VCs does not make a market. You also need to have customers and muscle.” Basically –

[00:18:33]JM: I also think there's value in having metadata management in, like, integrated into the rest of the workflow. As opposed to having a separate metadata management tool.

[00:18:46] RB: I think a separate – so yes, that's a great point that you bring, like having a separate metadata tool, and I was going to come to that, is just a bad idea. Because it's bidirectional, right? So when our customers use metadata, as they're building the business logic, we are computing lineage on it. Every time they commit lineage, we are on top of that create doing search. Now also, what our customers say is now, let's say you wanted to do data quality. So on the data set, on the metadata, they can go and put some rules. Saying this is blood pressure should be between 50 and 250. And now, what are you going to do?

Now, the rules need to get injected into the pipeline from the metadata to say, “These are the latest rules, do the data quality against these.” And then the results of that need to again go back to metadata, and there, you need to be able to report this is my data quality today, this is how it has done over the last one month, and this is the data quality across my 50 pipelines, and this is how we’ve, as a team work to improve it.

So one thing is part of the metadata is constructed while you're doing data, using local tools, partly from business logic, partly from the execution stats, partly – and then some other things like data quality gets generated from metadata and stored back into metadata. So metadata and your local data engineering tool are so closely coupled. You get a good tool, 90% of the metadata is already there. Every time you run a pipeline, the basic quality can just be put there. But if your metadata system is completely separate, how are you going – it's like, why do have it separated? And then they're going to run into other challenges. And then there are other dynamics of sales, right?

Because here, we're just talking technology. But thing is, that there is budget allocated for an ETL tool on prem. Now, they're like, let's move to the cloud for larger enterprises, and they're like, "Okay." And then you're like, "No, I want a metadata product." It's like, "Well, how did you do metadata before it?" The ETL tool provided the metadata. Okay. Does it still do that? Yeah. Why do you need the second metadata solutions? I don't know.

I think it's an inferior solution in general. I don't think much will come out of these startups. But I mean, good luck to them. They are my friends. I know a lot of people. I know the people from LinkedIn. I know a mass from Metaphor. I know [**inaudible 00:21:13**] and their friends, and I hope the best for them. I just think it's a new category creation, because metadata has not existed as a separate category.

[00:21:21] JM: So when you think about other problems that prevent people from having a successful data engineering experience on the cloud, what are the other main hurdles that you see?

[00:21:35] RB: I think the main hurdle, and the main hurdle, I would say is the lack of tools. If you go back. So I think there's a slightly deeper point there, and the basic thing is, you go to start building a data lake stack, right? And everybody is wrestling with infrastructure. There's a lot of toil, very little value coming out of it.

Let me give an analogy. So let's say when you were using Hadoop or Hive, you were like, the interview question would be, "Oh, are you a developer? Would you use a map join? Or would you use a shuffle join?" That's great. But why do you have to know that? Because the query optimizer doesn't know how to do its job. Once the query optimizer inside a Snowflake or a Databrick Spark, or a Spark gets good enough, then you don't need to worry because you will have your new query planning and they just plan the right query.

So basically, what's happening is that people are – today, when I go into organizations, somebody's setting up Spark, somebody's setting up Airflow, then you're setting up like some version of Presto. Just like in the Hadoop land, engineers wanted to get N different technologies on their resume. Now, they want to get N different cloud technologies on their resume. Then you look at it, and you say, "How does that help anybody?"

Basically, there has been this – I think there is just a misunderstanding of the direction to be taken, and misunderstandings can be like, Hadoop was a big misunderstanding, right? So a misunderstanding in Bay Area, in the technology industry can be 10 to 15 years long. I've said this before, when m MapReduce came out, Michael Stonebreaker, who's like the database researcher build and startup world is like, "That is a terrible idea." That is like, so backwards, right? But then the question is, why did it get built? It's because Google has a particular problem. They have an amazing cash cow in search, and they need to drive the cost of the stack down to zero.

So they're building of the stack is focused on cost, cost, cost, not on value, value, value, right? The value they're deriving from the search is the same. They're not saying per engineer value, per this. They have tons of engineers to throw at the problem, tons of resources, and they want best margins in search. They're going to still build MapReduce, though it is 20 years behind technology, or 30 years probably late. It might make sense for them. But then after a little bit, everybody realizes this.

Right now, what's going on, is there is this notion that I will assemble a data stack, and on top of that, I will start building some tools, some frameworks. I think everybody's stuck in this, that they assemble some stack together, then they assemble some framework on top of that together, try to get value out of it. Most people are not. That is not the stack if you want fast value from business data. Most people are stuck there.

This is a very interesting thing. So I talked to a product manager. She's here in Bay Area. She works for one of the barrier tech companies and she's like – I was discussing a product management position in Prophecy and discussion, she's like, "Oh, so this is for people who cannot look for this sort of people who don't have – who can't hire the kind of people that we have access to in the Bay Area." I'm like, "Okay, tell me more about your data ecosystem." And then she's like, "Yeah, we have all this data. The business analysts don't quite understand how to get it. This is always so much work. We have all this new data lying around. We know we can get business insights from it, and make improvements, but we just never get to it."

And after a while, I'm like, "Do you realize that you feel like you have all the talent in the world, and your data engineering is not quite working?" And then she talked about, "Yeah." It's not about time. So the basic problem is one, there is not like – this notion that you're going to use N different tools is a bad idea. People are going at stitching their own thing. Abstractions, that is a bad idea. And then most are just really struggling. So I've hardly seen any good working data engineering work. Before this, I was in motion. I've been doing this for like, 8 to 10 years now.

[00:26:00] JM: Okay, when you look at that reality, that there's not a lot of good data engineering practices in the valley. Do you think it's a lack of tooling? Because I mean, your system is built on Airflow and Databricks, I mean, there's a plethora of data tools out there. But is it a lack of usability in the data engineering tools? Is it lack of more prescribed workflows that are enabled by the UI?

[00:26:30] RB: Basically, yeah. I mean, we can go into two to three different things. So one part of it is what you said is there's a plethora of data tools, but they're not really – they're mostly processing engines, right? Airflow is a processing engine with a poor interface. It is like, how do you develop your Airflow code in Python? And then can you interactively test it? No. You deploy it, and then it's not deployed if it's not there.

So, yes, to your point, two main things. First, is there is a space for tools, Tableau works, Power BI works. So this notion that processing engines are enough. Then you would say, "Hey, why is there Tableau?" It's like, yes, there is a space for Snowflake, and then you need the rest of the stack on top of that. That's why Tableau – yes, now, of course, Salesforce bought it for 16 billion. That means so many users use it. First, is it is what is in the cloud is just the lower levels of the stack, and on top of that, people just program in the Bay Area companies, because they as developers understand what is the interface required for them.

Okay, the data analysts don't understand the value stuff. It's like the data governance, people don't know. Tough, right? Basically, open source and these barrier companies don't have design cultures. They have engineering cultures, and that's how they become like Google has engineering culture, LinkedIn has engineering culture, the usability and design of these products is really – this is really low, and then nobody's providing a complete solution. So what do I think is the right solution? When I say, what do we as a company believe? I believe you should use Databricks, use Prophecy on top, and that should be 90% of a data stack.

We are not there yet. We will have it in some time. But local development, search, lineage, deployment, cost management, performance management, everything should come out of one tool. There is nothing technically that that says you should have two tools. Or you should have two engines. Databricks can do it all. We can do it all. T's just so we'll get there in timing. But the second point you said, when you're using low code, now low code in Prophecy is different from old school ETL tools. Because old school ETL tools, you do visual drag and drop, I store it as an XML file. You are cut off from the whole ecosystem. What we are saying is no, no, no, no, no. You do low-code drag and drop development, you are visually modifying code. It is like you do the same for websites and web flow.

Basically, as you're visually doing it, you're getting high quality code on Git, well structured, very readable, very maintainable with high-test quality, with the build scripts. You have all of that on Git. Visually, you're writing code. Now, the question is, if you have Prophecy, and you're visually building something, and you hire a data engineer who's coding it in PI, Spark or Scala, what is going to be the quality? The quality of the Prophecy code is better. It is standardized, it is well structured. It is performant, and we've seen it in end customers and we have data for it. Then the next thing is, now what's going to happen when the next data engineer comes in, right? So usually, you write a few lines of Spark code. I mean, maybe we believe that the state of the art and tools is a notebook right? This Jupyter Notebook. That's like amazing tooling for developing data engineering.

But typically, people write two to three lines of code, another two to three lines, another two to three lines, run it, run it, run it, run it, and at the end of it, have a block. Then they put it in production. Somebody else is on, like I said, pager duty, and something breaks, and they can't understand the code. Somebody new comes in, they modify something, they don't understand what the restriction. There are no tests. They put it in production, something else breaks.

So this whole standardization is, I think, the top value in low code, I would say. Yes, you can make it accessible to a lot more users. But standardization, because once you have something standardized, you can run it at an industrial scale. One data engineer leaves, the next one comes in, that's okay. Because everybody can quickly understand what the data pipeline does. And everybody can cooperate. And then the thing is, is there anything you can do in code that you can't do in processing? No. You can create your own new visual components. We call them gems. So if you're a data platform team, you can create standards that you want people to use, reusable, think of it as reusable function libraries, and roll it out to the team.

Now, everybody who's a data analyst, you're like, “Okay, this is how I write to Presto. This is my high performance connector. And then here's the simple UI to use it.” And everybody uses it. Why does somebody need to go and write that code every time? So the basic thing that's saying is low code leads to standardization. Standardization leads to speed. Standardization leads to being able to deliver data, to machine learning, to business intelligence, at industrial scale, with industrial speed. It's like, this is low code, every component has a test, quickly, you go to production, you will run those set of tests. Everybody's running the same thing. Everybody can understand each other's code. So everybody's built out of the same Lego blocks that you know our code.

[00:32:03] JM: Are you totally prescriptive on Spark plus Airflow? I mean, you mentioned Presto there, can you imagine a world where you build UI elements for making Presto easier to write?

[00:32:15] RB: I mean, we could do that. One of the things we've thought through – right now, when you're doing visual drag and drop, we can generate either Scala or Pi Spark. We could also generate SQL ++, SQL with functions and references. Let’s say DBT. In Scala, we have SBT. That's a build tool. Let’s say, a DBT format, that’s just pure SQL. So we could do that. The question really is why? What is Presto better at Spark? I actually don't know the answer to this. But my basic instinct would be to say, “I don't know. Cost?” But what is Presto better than Spark at?

There's a couple of things. One is, what do we write to? So we can write to Presto, and we can write to Snowflake. We typically we process in Spark, and then write out to these systems for like ad hoc querying. But then, if you're using Databricks or Spark, do you really need to use this other? The question is, are we using in the industry in many of these tools, because we just want to use one more tool?

So that would be the other question, but to us, right now, we only run on Spark. And then for scheduling, we support airflow. Actually, what we are doing is, in about eight weeks from now, we are coming up with a newer release on Databricks jobs. Because we have some customers, which are, let's say, in the Fortune 10, and they are running tens of thousands of pipelines, and some of the complex use. And Airflow is a deal – we also support Airflow, and we have the same investor in inside venture. So we are kind of sister company, but there are use cases for Databricks jobs is better, right?

I think we're coming up with like prophecy for Databricks product, which is like Spark Delta Lake, Databricks jobs and no other dependencies. That's a really simple stack, right? It's like Databricks Prophecy on top, and that's it. I don't need anything else. I got my scheduling, I got my development, and Databricks jobs has been making good progress. And for us, it's like fewer dependencies and it just works, definitely. But to your point, for processing engine, it's only Spark and Delta. For scheduling, it’s Airflow and Databricks job soon. And then, we could tomorrow run on any data warehouse or SQL based systems. But we'll do one thing before we do too. As a startup, you realize this, we got to stay focused. We are going to narrowly focus on Spark for now. We could do it yeah.

[00:34:47] JM: Yeah. I'm sure we can go deep on Presto, but that's not really the focus of the conversation. I guess, one thing I'm curious about is since you've been talking a lot about how Bay Area companies, which is, I think, probably a leading indicator of a more general set of companies, how they do data engineering. You've touched on some of the ways that they could be more productive, and a lot of it probably comes down to workflow. A lot of it comes down to tool choice. But when I look at Prophecy, it's kind of a tool that I'm wondering who it serves exactly. Does it strictly serve the data engineer and just empowers the data engineer? Or is it something where a data scientist can get started with doing data engineering and ease into data engineering?

[00:35:41] RB: So first, I think that our top level view is, first is we are coming to this with the point and of course, that's the view from the company point of view, or how we look at things is, nobody should be writing a data engineering script. It is just a bad thing to do. It's like saying, “I am going to write a four loop in C, to join two tables.” No, there is SQL out there. What the hell are you doing? It's just a new generation of people who don't know. And then the next thing comes is who are the users?

Now, I think a lot of the thing that is happening in market is, products are getting built around functionality rather than a personality. So I would say, this is for whoever wants to do data engineering, and there are subtleties to it. For example, and I'll give you an analogy, just to switch gears. I do our design myself, and I've worked with a designer, I have a very good designer. Actually, he's a guy from Brazil, who sits in Bali, and I'm jealous every time I see him. Because he's always got the beach in the background.

So he works from there. I work with him closely. We use Figma. So Figma is used for design, not by the designer. Before that, we had Sketch, and Sketch was just for the designer. But Figma is like, it’s online, the product manager is there, the implementing engineer, UI engineer is there, the designer is there, whoever needs to do that, and there's multiple stakeholders in there. Now for data engineering, Vaughn is the data engineer, data engineer, maybe they were a visual ETL developer. They are super productive. If you're coding, and you can write Pi Spark scripts, et cetera, still, it is better than writing those scripts.

But that said, now, analysts can use it, data scientists can use it, when they're doing data engineering. Now, the other dynamic that we are seeing is the data platform teams are creating these some of our standard gems, the visuals, and giving it to the data analysts. And saying, “Here you go, don't bother me, don't come to me every time you need to build a data pipeline. Here you go, here's your self-serve platform.” Then they come and say, “Hey, I need this thing, and this thing, and this thing. These are my three problems, can you solve that?” The programming data engineer is like, “Let me create a visual component for you in Prophecy.” They create a visual component that does that thing and say, “Here you go.” Because they can plant that visual component in the middle of their pipeline, see the data going in, see that data going out, configure it a little bit, and they're good to go.

So I think the earlier case is like where our primary focus right now is on the data engineers, the visual developers, the coding data engineers, very much there. But I think, we are starting to see use with some data scientists and some data analysts when they're doing data engineering.

[00:38:31] JM: I can relate to your appreciation of Figma. I've certainly done a lot of work in Figma, and it's quite pleasurable to work in, quite easy even as a non-designer. And given that you have built something that's – given that we're in a time where you do have that level of design productivity, it kind of feels like you can iterate even faster when you can get to a point where the CEO can do design work, I mean, that shortens a lot of cycles there. So kind of interesting meta point.

[00:39:07] RB: There is a lot less iteration, because that's what we were saying. We wanted to do a quick UI overhaul, because we want to do a discrete Databricks, Prophecy for Databricks released. And I was like, if I hire a UX developer, and of course, I'm coming from product background, right? Then I'm like, I'm going to hire a UX designer, I'm going to have this, and then they're going to come up with something and iterate, iterate, iterate. It's going to take a year. In Figma, there is the designer, there is me, the product manager, and my co-founder **[inaudible 00:39:37]**, who also does product management, coding, and both of us do coding design and product management, all three, and our UI engineers. We are in there and we built the old design system, and it quite ended up being complex. It's simple to look at. But yeah, our needs are complex.

So we built a complex design system in about eight weeks and we're iterating on that, and in another eight weeks, we will have it all. Reskin UI, reskin, and there's no other way to get it done. Unless all the decision makers, the implementers, the designer are all in the same tool, and there is like Q&A streaming on the right hand side. Why can't we do that in data engineering? I think we don't have that today. But where we would like to get to is that you're building a data engineering pipeline, and you get stuck somewhere, and you can ask somebody who's a senior member in the team to just hop into your pipeline and help you out with a little piece. Or you could just ask, comment and say, "What does this mean?" So there's a lot of there's a long way up in terms of being more productive and having a good environment.

[00:40:43] JM: Yeah. So if we returned to the topic of Prophecy itself, and we've touched on a few elements of it, but I think some of the critical parts of it are these things that you get on top of kind of the run of the mill Spark experience. If you have search and lineage on top of your normal Spark experience, that's a great deal more productivity. Is it work intensive to build search and lineage on Spark? Maybe you could talk a little bit about how to facilitate that and what kind of engineering it's taken.

[00:41:18] RB: Sure. That's good. So I think, let me quickly touch on. When we say, let's say we go out with somebody who's a potential user, the things we are telling you is four things, and I'll just jump straight into what you're saying, because it'll flow from there. First, we say, "Hey, we are low code." Low code means easy to use, and standardized, and many different users can use it. That's great. The next thing we say is we are code based. That means code on Git, CI/CD, best software practices, both of these targeted at scale, industrial scale. Then we say we are complete. You've got search, you got linear, you've got **[inaudible 00:42:00]** solving your entire problem. And then the fourth thing we says we are extensible. Extensible means, of course, you can reuse subgraphs. You can reuse the same business logics, the same business rules, and you can reuse visual components.

Now, let's come to what it takes to build this right. So first is as you're doing visual development, we are doing cogeneration, and the code is neat, clean, all of that. Now, if you change the code, we pass it back to the visual graph, so you can make small edits. The core structure remains the same, you can change a few lines. Let's say your data did merge. You have visual structure one, visual structure two, you did a merge. Now, the code merge, so we'll extract back the visual structure. That was one point of complexity.

Now, the next point of complexity is, well, now you need to write your own visual components that we call gems. When you have to write your own gem, you're writing some sample Spark code and saying these things should come from the UI, and here's a small function saying how the UI is laid out. From that, we are generating the code generation. We are generating the parsing, and when you use your visual component, you will get autocomplete.

So that is a lot of compiler tech, and then we support multiple languages. There's tremendous amount of compiler tech. Every time you modify your code, we do a local Git commit for caching, and then we'll do the Git commit to your GitHub, after every few, periodically. As you're doing that, now, what we're doing is for each one of your visual components, we are going into the Spark’s logical plan and computing what the output means in terms of the inputs. We are computing lineage per visual block, then we are going and adding it up to the workflow, then that workflows writes to a data set that is in a different project.

Now, it's across workflows across projects. So there is a part of computing lineage from code, which is hard, because people can write anything. If you can do function calls, it’s like scholar Pi Spark code. Then the second part is serving it fast. Where you need to now – let’s say I look at a value, and I say – I mean, in this case, it might not make sense in terms of names, but let's just take that example because it's easy. Let's say I find out that hey, my full name is wrong. Then I go back and trace it and two pipelines earlier, it came from a first name and a last name and the CONCAT of that. Now, maybe the first name is wrong. Now, I need to change that.

So there is this merge and split of columns that is going on, and dynamically I need to be able to move that. I need to have it then – okay, so this is great. Right now I have based lineage, but now what is going, and I have like graph on top of it. First, is how do I even get into lineage? So I need to be able to make this entire lineage searchable. I have my Elastic Search and set up the right way, prism preprocessing done. Now, I get into lineage, now I need to be able to go left and right, and I'm doing this dynamic stitching from lineage or I'm pre computing it, that's fast graph accesses.

I think the development environment has a lot of technology, which is around compiler tag to make the visual thing work, and then there's also time constraints. For example, if you are in a visual graph, and you create a component, immediately, you want to see what the output schema without running it. So there is interpretation going on there. There's a lot of technology in the development environment, and then there is a lot of technology in computing column level lineage. Now one thing you can say is, “Okay, if you look at the, let's say, you said, okay, LinkedIn has DataHub, and Lyft build.” And all these companies build these things, and there are many startups funded on that, and they've been using it for years. You ask them, “Do you have column leveling?” Not one of them has it. That was the hard part. They're like, “Oh, we have data set lineage.” That's great. That's better than nothing. But that's not the hard part. The hard part is called, and they don't have that. The question is the ability to say no.

So what I'm saying is, there is some compiler work, there is some going inside this pathological plan, just for computing, and then there is the search, and then graph based serving at speed. One other thing, right? The other thing is versioning. That is very important. Basically, when I talk about production customers, they're like, “Okay, this column and back.” And they want to debug that. How do you debug that? So you go back from the bad column, and you say, “What was the last pipeline that wrote it?” Now I can see in the lineage, “Oh, last pipeline, it was passed through. The pipeline, before that, it was passed through. The pipeline before that, this thing wrote.” Okay, maybe this isn't my source of error, has anything changed in the last week? Because now you have this Git versioning integrated with lineage, now you can see juxtapose two different pipelines from two different times and say, “Nothing changed in the last week. This value went bad in the last week, maybe I need to go further up the pipeline.” And see where something changed.

I mean, there is integration. The lineage has to be integrated with Git and versioning for it to make any sense and be helpful for debugging, because I just don't want the current view of lineage, I want how lineage has been changing.

[00:47:30] JM: So, you've touched on a lot of interesting engineering subjects there. I think, particularly the auto code generation, doing low-code data engineering, and automatically generating the code from that, and then being able to reflect the opposite direction to where you write code, and it gets turned into low-code modules. You said, that's a lot of compiler wizardry, and we could go into that. You also talked about the column level lineage being more difficult than row level lineage, and that's another area that I'd be interested to dive into.

But maybe rather than going deeper into implementation of that stuff, people may be curious about that. But, I think, rather than going deeper into that, we should probably just talk a little bit more about actual data engineering at scale. I think one of the things that I'm curious about is sprawl of data engineering workflows, and how those workflows get organized within a company. How the different pipelines are getting managed, because at a really complex company, like Netflix, you have so many different data pipelines, I feel like things can get lost or things can get forgotten about or orphaned, and I just like to know your perspective on data engineering pipeline management.

[00:48:59] RB: That's a great question and a good direction to go. Because data engineering at scale poses completely different challenges than data engineering at small scale. So basically, the problem is first, like I said, standardization and having similar processes everywhere is super important. Because if everybody's kind of building their own network data stack inside every different business group, that becomes a challenge. The other thing is, you want to have – and cloud gives you great decoupled infrastructure. So basically, you don't need to share the same cluster, and the same scheduler. You can have N different versions of it.

But I think the main challenge, so when we look at data engineering at scale, I think the problem is like literally, within organizations, you have L1, L2, L3 support for your data pipelines. People will find some issue and file a bug and get an engineer on the phone to quickly fix the data pipeline. You have to worry about that. Is your production support productive? Can they quickly point to an error? Can they handle that? Then the next thing becomes is how do you manage your data pipelines? Right now, some organizations use a single Git. Every different team has a different folder, and how typically we Git 10 is like, in one of the folders, you start writing code with Prophecy, and the other ones are still good right? And then over time they adopt us.

But writing code in GitHub, I think there's no good way around it. Having GitHub as a source of truth, having versioning, having CI/CD working, that's there. The next thing is you need to have high test coverage. Very few people have it. The thing is, the person who wrote the pipeline is not the person who's converted it. After some point, somebody else is going to come and edit it. They don't know all the things that the first person thought. They're going to change something, a test has got to break and say, “No, this is not the right thing.” Then comes the ability to quickly go back to the previous version.

Okay, this pipeline just didn't work. Right now, for example, what we do is we say, “Okay, you tag this as 0.2 version. As soon as you tag that, we made that the live version.” Now, you realize that that version had a bug, go to the previous version, say that 0.21, tag it, and your schedules are to get auto deployed with the new artifacts. So, there is this amount of automation, where it's like, is your – so this is saying – I think I'm trying to put some good structure around it. One is, can you develop at scale? Develop at scale means a lot of people being productive. Develop at scale means standardize processes, and Git, and then now the thing is one to develop at scale, can you put it in production at scale? That means, is that a seamless process? If you make a mistake, and you push something back very quickly, and have the system correct itself.

And then the third big piece would be, do you understand what the hell is going on? Can you understand how the values are produced? Can you understand where your cost is going? That’s the other new thing that come up is people want reporting on top of their metadata. Right now, they don't have metadata. Just getting your performance stats in one place, they don't have it. But what I want to know is, okay, I have these 20,000 pipelines running, which one of them, the results are not even getting close? Then I want to say, which one of these processes really small amounts of data, but the cost is really high? And maybe that's where I should be, if I want to do cost management do it. It's like if I'm – so figuring out, which pipelines are starting to take much longer or something is going wrong.

Developing pipelines at scale requires standardization. Putting them in production at scale requires standardization, and we have to go towards like infrastructure as code, kind of thing. Code as the truth for what is deployed. This notion that you will go and deploy into Airflow, as a script, and then undeploy, and then something breaks. Then you say, “Okay, I have the state in my mind of what it was supposed to be, and I can see what it is.” That's something Terraform solved a long time ago. If you write what you want your schedule to be, do you want it to be enabled or not, and just push it to production.

So that production isolation at scale is not there. And then just understanding and insight, right? It's like my sales team has more insight than data engineering team in our top area company. The engineers in a data engineering company, you ask them, what is your data quality at? If he gave you two months, three months, how much cost could you shave off?

Again, of course, many times the data analyst, like the whole point of data engineering is to give the consumer, the data analyst, or the machine learning person understands data. Anybody running at scale, they have no idea. They're all guessing based on the column name. They have like 50 copies of each data set. Things just get lost. People can't figure out what data set is actually – user, a customer, you're going to find 20 copies of it.

[00:54:24] JM: Yeah. Well, as you begin to wind down, you've talked about a lot of different things. And I think, probably the driving point, that people who are building in the data engineering space would want to take away from this episode is, I guess, if you are looking for something that you can use to facilitate workflows around Spark and Airflow with some of these other features. Prophecy is obviously something you can use. And then beyond that, I'm curious where you're taking the company and what you see as kind of the next steps of what you're planning to build?

[00:55:08] RB: I mean, if you look at the company, what we are looking at is saying that the data processing engines are here, the primary problem is management of data, and getting value from data, and getting value fast from data, right? That's a problem everybody has, and most people are struggling with it. For us, it's like, there is so many problems to be solved for us just in data engineering. Right now, it's very much on – I think we are not doing something like, I won't say, grant. We are a very nuts and bolts company. We need to make sure everybody has good performance, that every time you run a job, and you can roll them up and understand them. We are so far from just those basics. Everybody has data quality on every data set to start with, and then they can go and do data, or improve the data quality.

So all of these things today, the very basics are not there. We think it will take us about two to three years, to get all these basics in place, and we develop very fast. We have a product that provides a lot of features, but there is just so much more to be built. Just like, Databricks has built a large stack, which kind of solves your complete problem. We are building a large stack on top that solves your complete problem. And then the problems to be solved are many. It's just like, there is no product that does that. That will keep us busy for years. I wish I had something great inspiring next thing to come. This basic thing, I would be very happy if a data engineering comes in and they say, look at Prophecy and they say, "Okay, I will not try to stretch my own data stack to gather with 10 different technologies, and then continue to wrestle with infrastructure. I will focus on my business problems."

Yes, the platform team can go help create some standards and enable people and focus on more value add activities, such as performance and cost and helping the users enabling more users in the company. I think we are very far from there, where BI and ML are fed with data fast and timely. The amount of business value that can be derived from that is just so tremendous for our customers. So we are we are very, very focused on that. I think for us, the other thing is we are just saying that the new low-code stack needs to be built on top of code? That's something is like how does that go all the way to business metadata to even an understanding – our customer is like, "Can you put the business meaning of this and propagate it on top of our lineage?” I write it once and then everybody who's connected to it knows what this business value means.

I kept saying, the value is just so much to a business. Right now, if you structure data, structured data infrastructure together, and then companies who want to build and aid the Bay Area companies and say, “We’ll become technology companies”, some new CTO will come in a large enterprise say, “Oh, we are going to do open source, we'll build everything on our own.” And then they'll start building a framework on top of Spark. We just want all of those projects to stop. It's not worth anybody's time. You won't build a Spark yourself. You won't build a Snowflake yourself. You won't build a Tableau yourself. It's like, stop building that stuff, we’ll build a much better version for all of you.

I think for us, it's just like data engineering has to work. And then, there's a lot of areas to expand, such as feature engineering, et cetera, and enabling them. I think we want to stop where BI starts or ML starts. But getting the first piece of data and you starting your BI dashboard, or your ML algorithm on TensorFlow or whatever, until that point, anything you need to do should be in one platform, completely visible. Everybody can use it, everybody can see what's going on. Completely lit up, so you can see all your pipelines, what's working, what's not working, where the cost is going, follow any pipeline, quickly seeing something moved to production change. That's a very, very challenging thing to build a very heavy lift. So I think we're going to do that.

[00:59:35] JM: Well, I think it's a really cool project. Obviously, I've been following pretty closely and congrats on the series A, obviously, and I look forward to following it in the future.

[00:59:48] RB: Thank you so much. I think your last podcast did help get us in front of some investors. We love that. I think we are with some exciting times ahead and I'd love to come back and talk to you after a few months, once we make progress and share interesting stuff too. Thanks for having me.

[00:59:56] JM: Thank Raj.

[END]