EPISODE 1436

[INTRODUCTION]

[00:00:00] JM: Customer data pipelines power the back end of many successful web platforms. In a customer data pipeline, the data is collected from sources such as mobile apps and cloud SaaS tools, transformed and managed using data engineering, stored in data warehouses and piped to analytics, advertising platforms and data infrastructure.

RudderStack is an open source customer data pipeline system that pulls together this disparate functionality. In a previous episode, we covered the basics of RudderStack. In today's show, we dive deeper into the engineering of RudderStack with returning guest, CEO, Soumyadeb Mitra.

[INTERVIEW]

[00:00:33] JM: Soumya, Welcome back to the show.

[00:00:37] SM: Thanks, Jeffrey. Really excited to be here again.

[00:00:39] JM: Yes. Last time, we did an overview of the customer data landscape and what you've built with RudderStack. And today, I'd like to go a little bit deeper into the engineering and some of the details of what you've been up to since we last spoke. So, just to refresh people, RudderStack is integration point for customer data, basically. Your different events and your mobile applications and in your SaaS applications, and your data warehouse and all kinds of triggers, that can occur around things like customer events, customer buying patterns, things like that.

The last time we spoke, you talked about essentially building the nuts and bolts of the platform. I guess to start off, once you get beyond logging, and analytics for the basic customer data, what kinds of features do you build on top of the basic core nuts and bolts of a customer data platform?

[00:01:49] SM: Yeah, and that's a great question. We have learned a lot since we last spoke around use cases that our customers are using RudderStack for. So, maybe before I give you like a direct answer to that, I'll give you a high-level understanding of like how people are using RudderStack. And then I can even dive into like, what are the things we enable? And what are the things our customers are doing? If that's okay.

[00:02:11] JM: Sure.

[00:02:14] SM: So, when we look at our customer's journey with their customer data, we kind of see like four steps. The simplest is where you are collecting the data and sending data, the same thing that you mentioned, like things from your app, your website, and then so on. You want to send it to different downstream tools that you want to send it to amplitude for analytics, or Mixpanel. And you want to send it to Google Analytics for marketing analytics, and you want to send it to like something like Braves or customer for running campaigns, right? That's kind of the first use case. You just want to collect the data and send it to different destinations. And that's what like the segment pioneered that space, and that's where every company starts.

If you're a startup, we see like a lot of adoption on less than 10 people company, you're just getting started with your data stack, you start there. Then as you grow, that's when you – let's say you hired your first data analyst, and you want to do more advanced reporting, then what is possible out of the box in Amplitude, or Mixpanel, or any of those tools. Maybe you want to bring some other data, like you want to bring your transaction data and combine that with even data and look at customer journeys based on that.

So, anything which involves more complex reporting, or multiple data sources, you realize that your existing tools are not enough. So, you want to get the data into a data warehouse, like Snowflake, BigQuery, Redshift or whatever, and you connect some some kind of a Tableau or Looker or any of those dashboards. And you build out reports on top. You had your data analysts to build that. So, that is where you go next once you have your first data analyst.

Now, as you mature even further, you start building some interesting active applications. It's not just about reporting, but things like let's say lead scoring, and you got all activity data, you got your CRM data into your data warehouse, you build out a report, but now you want to take the

score, unless you want to assign scores to users of lead saying that, are they active on the product? Are they not active on the product? And you want to assess push that score back into Salesforce so that your sales team can like prioritize based on that. This is a very important thing for product lead growth companies because like you want to prioritize leads based on what are they doing on the app. That is the third stage where you collect the data, you're doing some basic analysis and activating that data. So, that's what you see this category called reverse ETL and so on, and that's what we also kind of enable.

Further down comes the machine learning standard. It's not just about a simple lead score, which is how many times they have logged in, but you want to build like a more of a complex machine learning application. It could be like the same use cases, but like ML model for lead scoring. ML model for turn scores, and you want to push it back into the same set of tools, right? Salesforce or like Churn if you want push it again aside. So, that's kind of the fourth evolution of the stack.

And finally, you go to real time applications, right? So, it's not just about like pushing a score back into a tool, which somebody else follows up later. But you want to like in real time, personalize your app experience based on what they are doing in the app itself. So, you want to like do the same kind of ML models, but you want to sync that like data into some kind of a key value store so that now you can consume it in the app. You see things around feature store, and so on. That's kind of like the journey, we have seen our customers take through their customer data. The way we think about the platform today is like, okay, we have to enable that journey. There are things we have built today, which are like more of the plumbing layers, get the data into the warehouse, send it to the destinations, take the data out of the warehouse, send it to the destinations, and so on. And then there are other things which people are doing outside of the platform, transformations on the warehouse. So, that's something we're going to launch very soon. Instead of you having to write those DBT roles, and so on, can we do that? Comes further down, we will hopefully, at some point, build more interesting applications. But like, yeah, I mean, we haven't built the entire thing. But that's kind of the vision of like enabling these use cases.

[00:06:13] JM: You mentioned a few important pieces of tooling there, data warehouse and DBT. I'd like to get a sense of how the customer data platform integrates with the different

pieces of a data platform. So, can you give me an overview of the most important ancillary tools that connect to – or that see users connecting to RudderStack?

[00:06:47] SM: Yeah, that's a great question. And again, Ill kind of map that question to the journey that I was talking about, that customers take. At the very initial level, you don't have a warehouse, you're just connecting to all these different tools too, like SaaS tools, if you will, to generate your report. So, that's why we see like Google Analytics and Amplitude and Mixpanel and so on. Now, have enough visibility into which tools are being used and so on.

But then, the very next thing is you have now more complex customer data processing requirements, and you connect to data warehouse. That is the first most important tool. If I look at like popular destinations, I mean, Google Analytics is probably the most popular one. And then there are one or two other SaaS tools, but then the right after that is a data warehouse. So, any mature company has a data warehouse. By mature, I mean, even like companies sized – as long as you have a data analyst who can do something with the data, you have a data warehouse. That's the most important piece of technology we have kind of seen. And I think like we have also seen, like, you have seen that option of warehouses in general and that reflects in the stock prices.

Now, once you have the data in the data warehouse, we see two kinds of – you need to do some transformations, and you need to visualize the data. Analysis is the most first important use case. So, transformations traditionally used to be done in like something like DBT, used to be done in SQL. Write a bunch of SQL jobs and then a lot of that is moving to a DBT. Same with our customers, like I mean, a lot of our customers are just doing it in SQL, cron jobs and so on. But then more and more we see DBT adoption. And then the next step is the visualization layer. I mean, you see a lot of like, Looker is popular, Tableau is popular, a bunch of other BI tools, open source BI tools also, right?

So, if you stack rank, like what is the most important tool? Data warehouse, followed by a BI tool, and then probably like DBT, in terms of data processing. And then there is the other ecosystem, which we see which is slightly more mature companies, which is where you're not dumping the data into a warehouse, but you're dumping the data into some kind of a data lake, whether like S3 delta lake from database, or like some of the cloud storage, and you're doing

like processing in Spark. You rewrote some jobs to the synchronous transformations that you do in SQL, you can do it in Spark and Spark can let you do more ML kind of stuff also. So, that's the other parallel almost thread we see with our customers. And our customers are doing both. Dump into SQL, dump into a data lake as well.

[00:09:12] JM: How does the actual data movement take place in terms of like, if you're going from the data warehouse to and from the various sources and syncs, if you think about the data warehouses, the place where all this data is getting stored, all the customer data is getting stored, when are different transformations and algorithms being run? Are they being done as like triggered events? Or are they being done as batch jobs? Give me more of a sense of when RudderStack is being employed for things like transformations and algorithms?

[00:09:58] SM: Yeah, it's mostly batch. So, the typical flow is like, I'll take that lead scoring example, where you want to score your Salesforce leads based on the activity, they're doing in the app. So, you need your Salesforce data about leads, and then you also need the product activity data, which comes from your website, and so on. They'll deploy, again, a shameless plug for RudderStack, but yeah, I mean, then deploy something like RudderStack to pull all that data into the warehouse. And then that data is being pulled in like pseudo real time, right? The CRM data, Salesforce data, which you traditionally call it ELT is being pulled less than six hours. The event stream data that is slightly more real time, I mean, we can talk as frequently as five minutes, but it really doesn't matter for this reason, you can dump every half an hour.

So, whenever the data lands into the warehouse, that can trigger, like after you land the data, you run a cron job, which triggers whether it's in DBT model or a SQL model, and we have hooks to do that. I mean, API's can call into and wait for it. Or you can just monitor your data warehouse if it landed or not. So, you run a transformation, which will trigger something like a lead scoring, and you compute a score based on some function. And then once you have the score, you have to push it back into Salesforce. So, that is again, you can trigger it through an API, but it's often also set up as a batch job. You run the data, you run the transformations and push it back. So, we are doing the plumbing layer, the transformation generally is happening either in like airflow or, or crontab, or DVD cloud, or like whatever people tools they have.

[00:11:33] JM: I'd like to get into the engineering of RudderStack. So, first question I have around that is, how do you choose what programming language to use for the middleware of shuttling data from one point to another?

[00:11:48] SM: Yeah, I mean, we went to Golang. All our code is open source, like it's all written in Go. Why did we pick Go? I wrote a lot of the initial code as well. I think it was just that a lot of this, and I come from that background. I mean, I wrote like data systems before. Go was a popular language for like, very have a lot of multi-threading, and a lot of asynchronous events. There could have been a better choice, but I think like Go has been pretty good for us. And you don't get into like Java garbage collection problems, and so on. I think, that's why we picked go, and then we kind of stuck with that after that. So, there is some part of the code which is in JavaScript. We have these transformations, we call it, which take an event and like map it to the destination format. So, if you're sending an event from RudderStack or something, again, I have is a very good answer. But that's what it is.

[00:12:43] JM: Is there a difference in how the different data warehouses are performing when being used as the back end for customer data storage?

[00:12:55] SM: Yeah, it's a tricky question. I mean, we want to be like neutral to all the warehouses. So, I don't want to pick one over other. But I can give you a high-level view of what people like about the warehouses. I think what people like about BigQuery are two things. One is the pricing model. But particularly when you're small, I mean, you just pay as you pay for the amount of data scan. So, you don't have a lot of queries, you just click. You don't have to pay for storage, you just click on the number of queries. And the other thing that people really like about BigQuery is the ML functionality of that. I mean, like BigQuery has BigQuery ML, and we do see some adoption of the BigQuery ML. It can easily train an ML model in SQL. You don't have to set up that complex ML stack. I mean, we have customers who have built churn models within like couple of dates, and then if you have something working end to end.

So, that's where BigQuery really shines. Snowflake is the big guy here. I mean, in the mid-market enterprise, like Snowflake's pricing model really options. And of course, like the product is great. So, no doubt about this. I think that the main thing that Snowflake did was the decoupling of storage and compute, so that you can pay for them and scale them

independently. And that really sought their huge adoption. And we are like on Snowflake, because again, for the same reason. Redshift, I guess, I mean, I see a lot of recent releases from Redshift. They are the early guys in this space. They innovated the Cloud Data Warehouse, hopefully they will catch up.

[00:14:19] JM: Are the challenges that you've been facing as the company has gotten bigger, more rooted in engineering or in company structure?

[00:14:29] SM: Yeah, that's a good question. I mean, we definitely have a lot of engineering challenges. As we are scaling this faster, our system has been very reliable. So, that is a thing, like kudos to the team to do that. I think that's where we have done well. There is a lot of scope for optimization, I think, across the stack. I mean, we are paying Amazon, like what we should not be paying. That's kind of one thing. And we are a final integration business. Every week, there is something which is changing, and so are we catching up with that, is some work.

I don't feel concerned about it right now. We just announced a series B funding today. So, I think now it's more about companies scaling. We went from like, last time when we spoke spoke, we're like 30 people. I knew everyone. I mean, I still I know everyone, but like, today, we are 120 people and like scaling rapidly. So, thinking the organization, and this is not something I've done ever of this scale and managed companies of this scale. So, I don't know what I don't know. But I think that is the big thing for us to scale. Hiring is a very, very big problem. I'm hiring great people.

[00:15:41] JM: As far as what those engineers are getting allocated to, is there a lot of work on integrations? Or is it like scaling the infrastructure to deal with all the tenancy of the hosted platform? Or where are the most acute engineering challenges today?

[00:15:57] SM: Yeah, that's a good question. I think, we have organized engineering into pods. If you ask me, what are the hardest problems we have to solve? Integration is a big problem. I would say, it's a problem like automating how do you test that something is not going to break and so on. We have to build quite a bit of infrastructure, but we have to do a lot more. So, I think that's definitely a challenge.

On the core backend, there is a lot of room to scale that and like and make it like we're working on, like how to make it truly multi-tenant and like, improve efficiency by like, 5x 10x, maybe. So, they're hard engineering problems. But I think, we have good liquid engineers. We will figure it out. So, if you ask me, like, this is more of a solved problem. People have built high performance distributed systems. We will do it. On the integration side, there are a lot of unknowns. Depending on the destination site, and and so on, we have, like, what integrated with hundreds of tools? I don't think I haven't built any system like that, which can test those integrations automatically and so on. I mean, that's something we have to go and innovate, I guess. We have to figure that out.

[00:17:07] JM: There's been a growth in data middleware tools over the last three, four years. And I guess I'd like to get a sense of how you see RudderStack in comparison to other tools like Hightouch or Airbyte these other platforms for data movement. Is there a place for like multiple different data movement tools within a stack? Or do you think that each of these tools gradually expands to capturing more functionality?

[00:17:44] SM: All of the tools you mentioned, have raised to kind of money, and then so on. So, eventually, I believe there's only so much value you can get out of data integration. You cannot get enormous, huge deals on data integration. So, they will all go up the stack, I believe, in some sense. I mean, that's kind of my outside impression. As far as we are concerned, I think like, we are very much focused on customer data. We will solve not just the integration problem for customer data, but also try to go up the stack, like do more things around that, build what we call the platform layer, and so on. But it's hard to say how others think about it. But I guess like they have to.

[00:18:24] JM: Keep tell me more about what is in that platform layer?

[00:18:28] SM: Yeah, think about what happens, like the same stack I was talking about. You got your raw data into a data warehouse, then you have to do some basic transformations to combine multiple sources of data, and create – think of this as like a clean view of a user table, like in the marketing terminology is now customer physics. All of of CDB stock, whatever. A fundamental level, it is like for every user have one row with all the data you have about that user. I mean, some of those attributes to the user provided their age, sex, location. Some of

those are computed attributes. You have how many times have they'd logged in? How many times they have they done something?

Think of this like you create this customer view. And then once you have that customer view, then you want to do multiple things of that. I mean, at one end, you want to just connect to a reporting tool and build funnels and build reports on top. Then the other use case activation, you want to slice, dice that list and say give me all the customers who have done X but not Y. We have come to the checkout page, we dropped off, and you want to take the output and send it to different destinations. Or you want to connect an ML system. You want to train an ML model on top of that customer table.

So, the very next logical step, which like our customers are already doing is to create that customer view and then finally build those application layers on top. I mean, we think some part of that will be done by RudderStack, like at least creating the customer view and how do we make it easy for the customers. But then, some other things will be built outside of JavaScript. What we will own is the customer, like we define in the customer table that this is how it should be scored. And then other people can build interesting applications, whether it's analytics, whether it's the ML, maybe we'll build some others. That's how we kind of think about this space.

[00:20:12] JM: When you look at the data integration stack, the data integration, I guess, landscape of different tools, it's so much better than it was, say, five years, 10 years ago, when data integration meant something so different. I mean, it was more like, you know, if you want something meaningful from your data, you have to go ask the Hadoop engineer to get something for you. Now, it's some large scale, time consuming batch job. Whereas now it's much more real time and seamless, and driven by high-level tools. Are there any significant pain points today? Where are the pain points in the data engineering stack today that you think will be much more ameliorated in maybe another five or 10 years?

[00:21:09] SM: Yeah, that's a good question. I have to think for a good response here. Integration has definitely become much simpler, right? I mean, all the way from like – because the tools have evolved, the destinations have standardized. Now, you're not writing to some esoteric vertical, or something that you're dumping into, like one of the cloud data warehouses.

The dark sector has standardized quite a bit. If that is kind of solved, then I believe there's still challenges around the whole metadata around data, right? I mean, you're running all these pipelines and putting the data into a data warehouse. But then you have like so many different kinds of consumers of that data. You have analysts, to ML practitioners, to like to RBI, to like business users and it's a mess.

I think a good example is if you have a metrical lesson revenue, what does revenue mean? Even for a company of our scale, which is small, we have like subscription revenue, and we have annual revenues and everything is dumped into separate tables. How do you clean that up? There is revenue in the event data also, what is the clean definition of revenue? We have seen innovation on that space, like there are metrics companies, and so on. But I think there is definitely a scope to innovate around metadata and data sharing and data catalog and the broad thing. I think data quality also falls under this. I mean, I do see like, every time a dashboard breaks, our customers up to firefighter. Is it another stack problem? Did they release an app, which changed the instrumentation of the event? Now, if you have released an app, how do you go back? When you have a broken dashboard, how can you detect those things early enough? So, I think, yeah, in general, like pipelining is maybe solved, but making sense of the data is still hard enough problem.

[00:23:05] JM: So, you mentioned data catalog there. How do you see, when you look at the larger companies that you work with? You don't have to name names, but you have a lot of large companies that use RudderStack. A comprehensive data catalog at a company like Stripe, would be enormous. And it seems like there would be so much work around continually indexing that data catalog, and it seems like there's opportunity for tools to integrate directly with the data catalog, rather than letting the data catalog sync, just with the data warehouse. Do you think that's a viable strategy to have your tools like RudderStack write to the data catalog? Or do you think it just makes sense to defer that to further down the pipeline and just let the data warehouse sync with the data catalog?

[00:23:57] SM: It has to be like the tools like RudderStack has send the data, because we have lot more semantics that it's hard to get out of a data warehouse. So, I'll give you a very concrete example. Events are going to RudderStack, we are dumping that events into separate tables, let's say warehouse. Every event goes to a different table. Now, we know that like this event is a

production, people have set up things. There is a production event is like a test event and then so on. So, we have kind of that metadata, because that's how people have set it up on RudderStack.

So, passing this information to a data catalog is super valuable, as opposed to like, like the data catalog now having to try to figure that out from the warehouse. That's kind of one example. What other examples can I think of? In general, we do have metadata around our events. Another good example would be like, we're trying to standardize event generation. So, we know the tracking plans, we know what events are supposed to be generated. We have a standardized set of like ecommerce events and if you're an ecommerce company, these are all the events that need to be signing. So, having that metadata information, I think is super valuable for a catalog, which is hard to otherwise get purely by scanning tables in a warehouse.

[00:25:10] JM: How have you seen the open source nature of RudderStack be advantageous to you? Have you seen significant contributions back from the larger companies that are writing their own custom code around the customer data platform?

[00:25:26] SM: Yeah. I mean, that hast been another learning, and part of it is probably our execution too. Open source has not been as successful in terms of contribution back as I would have hoped for, right? Maybe it's because people don't like writing integrations. The other part of it is, even on the adoption side, we launched open source product, and the same time we launched the cloud product and so on. And then we have a very generous feed here. And then we have seen, like open source getting adopted, but cloud also getting adopted, and cloud adoption is much faster than open source adoption. So, it has been interesting learning. I mean, when I started RudderStack, I assumed that open source is extremely important to get to the data persona we want to target, but doesn't seem to be. But it has been like super helpful. I mean, we are still very committed to open source. I think we have created a community which is the pressure about open source and so on. But when it comes to just broader adoption, and so on, I think cloud has definitely overtaken open source.

[00:26:28] JM: When you see a customer adopt RudderStack, initially, for the customer data platform, what are the most direct, I guess, expansions that they typically are going for after? Once they start using RudderStack to record their customer data events, is there a kind of a

next place along the path that they're typically going? Whether it's reverse ETL, or event streaming, or using machine learning, like your built-in machine learning stuff, what's the typical next step after they begin recording events?

[00:27:06] SM: Yeah. I'll refer back to the same set of steps, as initially talking about. So, once you start with very simple, sending into different destinations, then you connect the data warehouse as the next step. And after you do that, I think the next step is – the first use case for data warehouse is analytics. You connect something like Looker, or Tableau and so on. Then comes like more of an active use case of data. The machine learning could be simple rule based. For example, we send how many times has this person, we compute a lead score based on two parameters. How many times has the person logged into the dashboard, and what is the total event volume they have sent in the last seven days? It's a simple function, which computes these two and sends that scoring two destinations. So, it requires that simple computation, which is done in DBT, and reverse ETL, which we take care of. That is the next evolution.

On consumer companies, we also see a lot of segmentation use cases where you get the data into the warehouse, and you want to create audiences. Again, rule-based audiences. I want to create like all the cart drop offs, and I want to push them into some kind of a marketing tool. So, we see these like, depending on like, whether PLG, our consumer companies are like, there are these rule-based transformations or activations, using reverse ETL. Then comes more ML use cases, where the rule based now becomes ML, right? Instead of you writing and crafting. And then finally, we have few customers who have built real-time applications. But it is not just about syncing the data into a third-party tool, it is computing those features, if you will, or recommendations in a warehouse, syncing it to like a key value store like Redis, or like some other version of that using reverse ETL. And consuming that in the app for personalization. So, it's kind of like different companies are in different stages in that data stack.

[00:28:54] JM: I feel like as a data pipeline grows in complexity, it can be harder and harder for the entirety of the data team to have an understanding of what the span of that data pipeline looks like and where different things are going. Have you seen any mechanisms for effectively mapping out that data pipeline and giving people a holistic sense? Or is it just like, do people just draw you ML diagrams? Or is there any kind of like auto generated system for people being able to map out their data pipelines?

[00:29:29] SM: Yeah, you're just making a bit for **[inaudible 00:29:30]**. Firstly, you want your data pipeline in one place, like instead of five different tools and so on. And the other thing is, again, we launched this – well, actually, it's in beta. We are going to launch maybe in a week or two, this thing around the data pipelines in TerraForm. So, just like your infrastructure is in code, why shouldn't your data pipeline be in code? We have customers doing it. When it becomes complicated, you want to like check in your – you define the data pipeline in something like TerraForm, check it into gate, manage version control and so on. That's one thing I'm excited about. We have some customers who are already doing it, and so on. So hopefully, that kind of becomes the standard. But it is hard to do these things when you have five different data pipelines running in RudderStack, shameless plug.

[00:30:14] JM: As you mentioned data quality earlier, data quality is one failure that can occur along the engineering of a data pipeline. But you can also have things like drop packets and other kinds of problems that can emerge, they can lead to data loss or inconsistencies in your data. Have you built any technology around retries? Or detection of data loss within RudderStack? Or do you typically kind of defer to the end user to sort through data quality problems?

[00:30:52] SM: Yeah, we definitely have to build a lot of infrastructure retries, all the way from like your client side, your SDKs, your browser SDK, or your mobile App SDKs. There could be failures anywhere along the way. So, we have retries on the client side, server itself has – we have high availability as the data comes to RudderStack and it confirms that, okay, that data has been persisted, only then the client deletes the data. And then now, RudderStack internally has retry mechanism and high availability on the data. So, if the destination is down, we can keep trying and so on.

So, we go a long way to try to make sure we never lose any data, even if another server goes down. I think, a big data loss is a solved problem. It's more merely an engineering problem. Data quality though goes beyond just click wire level data loss and data quality is a broader problem, because there are humans involved, right? I mean, you changed your app instrumentation, and you forgot to add an event or you moved an event. So, we have like alerts, what our customers do is like they'll connect to – we have very good integration into different

tools. So, you can connect to something like Grafana, and define alerts on your critical metric. If your revenue metric is what you care about, if that revenue metric is suddenly going down the number of events, then we will generate an alert.

We have kind of build tools around that. We have something called tracking plan. I mean, there are two ways you can broadly approach the, human aspect of it. Either you monitor through all kinds of – we have some basic monitoring, but you can connect to these end tools. The other is like you enforce. Enforcement could be at multiple levels. You can enforce on the client side and there are like tracking plan tools, which won't even let you write any events unless it conforms to the tracking plan. So, we have something called type writer. You can enforce on the client side, you can enforce at the router layer, at RudderStack layer, where we can say that unless the event conforms to the tracking plan, we will reject the event and will generate an alert. You can also enforce the data loading time like when you are loading the data into the warehouse or like reading the data back, you have tools like Great Expectation and so on, which enforces things when you're writing the data and reading the data. The human aspect could be enforced at anywhere, probably should be done everywhere. But it's still a pain. This one of the standard problems but, like to log into, say that is like we should not lose data. If you're losing data because of that is an engineering problem, but the stack that doesn't solve data quality.

[00:33:16] JM: On the topic of performance, so if you're logging a really high velocity of events, performance at the client level and at the server level are both pretty important. You've already kind of addressed the performance on the server side with Golang. On the client side, you have to use JavaScript, and getting high performance out of JavaScript can be a little bit tougher than getting performance out of Golang. What kinds of work around performance have you done to improve the client-side JavaScript stuff?

[00:33:47] SM: So, like performance means slightly different things on the server side and client side. On the server side, it is about even volume number of events we are processing. On the client side, it is running on that end user's browser, right? So, the amount of volume is very low. It's like how many things somebody has clicked and so on. The performance issue there or even on an Android SDK and so on is like less for mobile. But more on the JavaScript side is like the load time. If you're loading a big JavaScript, it slows down the website, and so on. That is the performance challenge, and we have done again, like, I don't know, there is a single thing that

we have done. It's like delayed loading, the high-level idea is that you load the bare minimum you need and then you load only other things which you need, if you need them, and so on. So, we've kind of done a bunch of engineering work around that. But I'm sure like we there is a lot of scope for improvement there and on improving like JavaScript, load performance, and so on. In fact, that is one of the value add of RudderStack. You don't have to load 30 different JavaScript, you can just load RudderStack and then we'll send it to everyone.

[00:34:44] JM: This is not the first company that you've built. I think your previous company exit through acquisition. Are there any points around market size or strategy that you can offer that have contributed to allow RudderStack to go further than your previous company?

[00:35:06] SM: Yeah, and this is a very personal opinion, and I'm sure others disagree. I think there are three risks you can take when you're starting a company, right? You have a technical risk, you have the market risk, and you have the team risk, at a very high level. Is the market there? Even build the technology, and then like, do you have the right team to go and build the technology? So, there are three risks we take. And different people, different founders have different skills. One thing I learned is I'm very bad at taking market risks. In general, engineers and engineers are like, very bad at first estimating a market, sometimes even selling and other skills around that. Then, as an engineer founder, you can take the other risk. We are good at coding, you can take the technical risk, and you build the right team to go and take the technical risk.

So again, I think what I've learned is the way to do that is, again, somebody like me, don't take a market risk. There are enough markets, which are proven markets, like CDP is a proven market. You have to bring a different angle. You cannot just say that, I'll do whatever else is doing. But like take a proven market, bring a better product into that market and execute well. So again, through this mechanism, you will not ever get an Uber, Uber innovated market. But it is like you want to match – you want to know what your strengths are.

[00:36:26] JM: Have there been any market risks that you didn't anticipate or market opportunities that you didn't anticipate?

[00:36:32] SM: We are really early in our journey. I think, so far, we have been very fortunate. A lot of things lined up for us, all the way from like, generally data warehouses exploding and not just in a real sense, but also in people understand that it's an enormous market. So, there's a lot of funding in the broader data space. That's kind of one thing that was held segment got acquired, and now they are distracted, with any acquisition, no fault of segment or failure, so on. So, I think a lot of things lined up. And again, we are very early in our journey. We have grown very fast. But I'm sure we will run into problems and every company runs into problems at some point. So, hopefully not this year, maybe a year after, but yeah. I mean, long way to say, so far everything has really worked out well. But who knows?

[00:37:18] JM: When you think about product scope, since there's so much that can be done with customer data, there seems to be a pretty important pressure on sequencing the product and allocating engineers to the right areas. How do you prioritize and figure out the right areas of the data stack to focus on?

[00:37:40] SM: That's a great question. If anything, I think that's one thing, we have not done a great job of. We already are fighting a big surface area with like three different pipelines, where there are specialized companies each with each one of them. But I believe that it's important for us to execute on our vision. And then we are doing on some of the transformation stuff. And so, we are already fighting a big surface area. We definitely need to do a better job. Thankfully, we have enough understanding from our customers like open source user, what is important what is not. But we need to do better here.

[00:38:11] JM: As you've moved to scale up the hosted platform, how has the deployment system changed, and your deployment medium for scaling the infrastructure that the transformations and data shuttling lives on?

[00:38:25] SM: Honestly, that hasn't really changed too much, and could be like part of our engineering thing. We were on Kubernetes almost from day one. The good thing is this is a very trivially, parallelizable thing. I'm processing events for your customers, there is an event ordering requirement, but it is only at an end user level. So, like let's say you are the end user in sitting in a browser, your events need to be delivered in order. Across users, across customers, there is no dependency between events. So, we can just like keep spinning up nodes which are

processing all this data. We are running an enormous Kubernetes cluster running hundreds of RudderStack nodes.

[00:39:03] JM: As far as that Kubernetes deployment, are you using a container management solution? Are you like managing your own Kubernetes on like EC2 clusters?

[00:39:15] SM: We are on multiple different zones, like we have European Center and like North America because of the data residency requirements.

[00:39:27] JM: Gotcha. On the machine learning side, what are the machine learning frameworks you use to drive the calculations on different machine learning algorithms?

[00:39:37] SM: Yeah, we don't have a product there yet. But we have worked with few customers to help them build some of the ML model. And we have kind of done both extremes, like the two main things to highlight is like we have done something's on BigQuery ML. Your data goes into BigQuery. And you literally write your ML model in BigQuery, right? You cannot tweet too many things, but you can select features in SQL and train a model and make it run periodically. So, you can literally have something up and running within a day. The other extreme we have done is like. We kind of do it on SageMaker, but I think that, Google I think is also has vortex. I mean, what we have worked with our customers on like SageMakers, Jupyter Notebooks, SageMaker.

[00:40:21] JM: gotcha. What's the biggest lesson that you've learned as you've gone from that 30-person team to a much larger team?

[00:40:32] SM: Big surprise was hiring is very hard and so on. I mean, I thought like, once you have money, you can hire people, but it's not, definitely. That was a big surprise. And big positive lesson was like, if the market is right, you get so much tailwind like when things are breaking left and right, but still customers show up and and so on. That was one of the pleasant things. Definitely different from like a lot of my previous startup experiences and so on.

[00:41:00] JM: Has there been anything that kind of breaks as you've scaled like communications wise?

[00:41:06] SM: Communication wise, within company communication, I think we have done a good job. Part of it is also because we were distributed first from day one. We have a team in India. We have a team in Greece, in the US. We have gone through COVID. Because of all of that, everything is on Slack. We are very transparent culture and so on. So, I think we have done a good job there. We also have like a strong communication, so everything is on Slack. We have every customer, we have a Slack channel. I think we can do better there because there, tracking becomes important. So, we have kind of built – not built, like integrated tooling around it, create ticketing on Slack, and so on. Customer communication, I think we can probably do better. But internal communication, nothing strikes to me. I think like we have done okay.

Slack, one thing I would say is Slack doesn't capture emotion. So, sometimes when you say something, if you don't mean it, it can come not in the right way on the other side. But I think that's like the culture you build. As long as people are trusting of each other, I think that's not a big problem.

[00:42:01] JM: You touched a little bit on the engineering of the middleware, just in terms of Go, I'd like to know a little bit more about how data gets batched within RudderStack versus Streamed. If you have a really, really high volume of events coming in, are you building queues of those events? And then shuttling them to different places? Or are you just moving them directly to the target?

[00:42:26] SM: I mean, we have to persist the data, because remember, like the destination will be down for some time, and so on. The highest level, the way to understand this, like any event that comes in is persisted. We built our own queueing system on top of Postgres. Because we did not want to ship Kafka with all the challenges with Kafka, and we wrote a blog about it before, like anyone wants to go into details. But high-level data comes in, we persist into a database and act to the client that you cannot delete the data. Then the next step is to transform that data, like we take the raw event, and depending on each destination has a different format. So, we have to convert that event into multiple destination formats. And then in between, we also support something called user transformations, where a user can also define a function that is executed on the event to click, remove fields, add fields, whatever. We have a powerful

transformation framework. So, that gets executed, then the event is transformed into all the different destination formats, and then the data is sent out

At a conceptual level, it's very simple, like data comes in, and then we persist it twice and send it out, then we build like a caching layer in between, so that you don't have to go and read and write from the disk every time and so on. That's kind of high level, one way to think about RudderStacks. All this code, all this code is open source for anyone who wants to like take a look. And even volume is not a problem. Because you can now parallelize data nodes. You can put a load balancer in front like which we do. We have a load balancer node and then events can – you still have to make sure that events of an end user, like on a given browser goes to the same node. But across users, you don't have any dependencies. So, you can spin up multiple nodes and just send events to all of those.

[00:43:59] JM: What's your caching infrastructure built in?

[00:44:01] SM: It's all written in Go. We build our own stuff. It's like in memory. It's not like anything sophisticated.

[00:44:10] JM: Gotcha. What was the motivation for building the persisting infrastructure in Postgres versus, like you said, Kafka?

[00:44:19] SM: The biggest one was like, I mean, from my past experience, I've worked with Postgre for a very long time. Even during my grad school, I looked at Postgres. We did a bunch of work from Postgres. The familiarity with that and like stability of the system, right? We built our queuing engine on top of Postgres, and I've never seen – we're processing like billions of events and across multiple nodes running Postgres. I've never seen a Postgres failure. It's just this kind of an amazing piece of engineering.

So, that's kind of one thing versus, I am sure Kafka has matured quite a bit, but when I was initially managing Kafka clusters in my previous job, like 2014, '15 timeframe, it was like a big project. So, we needed to have – managing was a was a big pain. And then when we designed, RudderStack was designed to run on customer's environment, not just like our cloud environment. I want anyone to deploy and RudderStack on their own Kubernetes cluster, we

have an open source project. Because of, I guess, my bias and so on, we don't want to ship Kafka.

So, that was one reason. And then the other thing is it's a slightly more technical details, but we wanted the data semantics of producer, consumers and so on, doesn't work really well, when you have like, huge multiplexing, and even retries and so on. I mean, we wrote a blog about it. I can get into the details, but that's the high-level idea. If you have to remove an event, and if it fails, you have to queue it back and not lose a position in the queue and so on. It's can be done on Kafka, but requires a lot of weird handling.

[00:45:44] JM: Well, Soumyadeb, thank you for coming back on the show. It's always a pleasure to talk to you and I'm really impressed with the progress you've made at RudderStack.

[00:45:49] SM: Yeah, definitely. Thanks again for inviting me. I'm truly humbled to be here. It’s an honor. It was great chatting with you, too. As always.

[00:45:53] JM: Thank you so much.

[END]