EPISODE 1073

[INTRODUCTION]

[00:00:00] JM: Customer data infrastructure is a type of tool for saving analytics and information about your customers. The company that is best known in this category is Segment, which is a very popular API company. The customer data is used for making all kinds of decisions around product roadmap, pricing and design.

RudderStack is a company built around open source customer data infrastructure. RudderStack can be self-hosted, allowing users to deploy it to their own servers and manage their data however they please.

Soumyadeb Mitra is the creator of RudderStack and he joins the show to talk about the space of customer data infrastructure as well as his own company, RuderStack.

If you have an idea for an episode, whether it's a company or a project that you're working on, you can go to softwaredaily.com and submit a topic. We're always looking for good show ideas and you can also support the show by becoming a subscriber and getting ad-free episodes and an RSS feed that has all of our back catalog, all more than 1,300 episodes, and it would support us.

Thank you for listening.

[SPONSOR MESSAGE]

[00:01:14] JM: Scaling a SQL cluster has historically been a difficult task. CockroachDB makes scaling your relational database much easier. CockroachDB is a distributed SQL database that makes it simple to build resilient, scalable applications quickly. CockroachDB is Postgres compatible, giving the same familiar SQL interface that database developers have used for years.

But unlike older databases, scaling with CockroachDB is handled within the database itself so you don't need to manage shards from your client application. Because the data is distributed, you won't lose data if a machine or data center goes down. CockroachDB is resilient and adaptable to any environment. You can host it on-prem, you can run it in a hybrid cloud and you can even deploy it across multiple clouds.

Some of the world's largest banks and massive online retailers and popular gaming platforms and developers from companies of all sizes trust CockroachDB with their most critical data. Sign up for a free 30-day trial and get a free T-shirt at cockroachlabs.com/sedaily.

Thanks to Cockroach Labs for being a sponsor, and nice work with CockroachDB.

[INTERVIEW]

[00:02:37] JM: Soumya Mitra, welcome to the show.

[00:02:40] SM: Thank you. Glad to be here.

[00:02:41] JM: The customer data infrastructure space, this space started with the company, Segment. Explain what customer data infrastructure is.

[00:02:50] SM: Think of customers data infrastructure as a platform. It helps companies collect first-party data from all the customer touch points, like whether it's a mobile app, or whether it's a website, or like a live chat, or even backend systems. Collect all those customer touch points and activities and send it to all the downstream applications where this is needed.

For example, you may want to send that data to Amplitude or some kind of an analytics product so that you can do like product analytics. You may want to send that data to some kind of a marketing on a different system, like MailChimp, so that you can like trigger email campaigns based on customer activities. For example, if somebody did something and you want to send them a coupon, like you set up a campaign in MailChimp. But to be able to do that, you need to send that triggered event into MailChimp. Similarly, you want to send it to Facebook to run some kind of a retargeting campaign. As a company scale up, you also want to get that data into your

own data warehouse. If you have your own recommendation system, you want to like get that event stream into that, right?

What happened was companies were like integrating all these different SDKs. I mean, for sending to Facebook, you have an SDK. For sending to MailChimp, you have another SDK and so on and you also have to build your own backend infrastructure to collect all these data and dump into your warehouse and so on, right? This requires substantial engineering effort. If you are to add in new destination, then you have to go and update the app. They're integrated a new SDK and like figure out what are the even semantics. It is kind of a pain, right? I mean, engineers never love wanting to do that.

It is kind of a brilliant idea from Segment that they kind of found this pain point, and the architecture is instead of sending to all these hundreds of places, you just send it to segment, and segment can fan it out to all these places. If you are to add a new destination, it's literally you go to the Segment dashboard and like click on that destination and [inaudible 00:04:49] to that.

Overtime, they also build like cool features, like you can replay events [inaudible 00:04:55] send it to a new destination and so on. That's like this whole space is called customer data infrastructure, the infrastructure to manage like customer event data. Now once we have that data, now we can also do like other interesting stuff with that. I mean, you can like create audiences and then so on, but that's like subsequent, that happen later.

[00:05:15] JM: Sure. Give a description for how CDI, customer data infrastructure, fits into the usage of a company. If I'm Uber, my customers all have mobile apps. I've got my backend infrastructure. How is this customer data infrastructure abstraction fitting into my workflow?

[00:05:37] SM: Yeah. In that case, what you do is you'd embed the SDK, the RudderStack SDK or your Segment SDK into your mobile app, and you will generate events that you care about. I mean, if somebody booked a ride, that's an event. Somebody searched for a place, that's an event. You instrument your code to generate all these events that you care about and then you send all these events through SDK, or the RudderStack, or the segment SDK. You send it to the backend, then you decide where all you want to forward those events.

You may only want some events to go to Facebook and only some of the events to go to like MailChimp, but you may want all the events in your own S3 data warehouse data lake. That's kind of what like the customer data infrastructure takes care of. It also defines the event semantics. RudderStack and Segment defined the structure of the event that you can generate. This is event type you can do and these are the properties that should be part of the event.

Then there is a mapping. I mean, how that structure gets mapped to the event structure that Facebook can consume. Segment kind of did that mapping. RudderStack, we are compatible with segment. I think segment did a great job around standardizing the event structure that pretty much like all companies will follow now.

[00:06:56] JM: Okay. Very cool. Segment was the first player in the market to come out. Other players have come out since then. Why do people choose platforms other than Segment?

[00:07:11] SM: Yeah. Segment is a great product. I mean, I've used it, loved it. There are couple of challenges with like the Segment approach, right? First is, as a data engineer, I wanted to deploy something inside my VPC. I mean, the very standard use case is I want to collect all these events. Only a fraction of those events I want to send to cloud, Facebook, or whatever, but I want to collect everything else and dump into my own DataLake, S3, data warehouse, and I want to do analytics on top, right? That's the pretty standard use case.

In that case, it kind of felt weird to now send this entire event stream to a cloud application to get a dump into your data warehouse, right? I would have rather hosted something myself. Increasingly because of the privacy and security regulations, I mean, you don't want to send your event stream, like cloud applications, particularly if you have like PII embedded in those events. That was the first thing. I mean, like a software layer which you could deploy inside your own environment.

The other problem that I found with Segment is like combining multiple data sources. Segment is great for capturing event. You do all the events, you send it to Segment, and they give you like nice tooling on top to create audience segments and you can set up rules that give me all

the people who did event X, but not event Y. Then you can do activation around that. You can send these to some cloud destination.

Now, that is enough if you are a small company, but any large, mid to large enterprise, you want to combine the event stream data with other data that you have. This is a firsthand problem I faced in my previous job [inaudible 00:08:50], where I was trying to put together customer data platform. We had the event stream data coming from the apps and so on, but we also had our backend internal billing system, and we also had our backend internal CRM.

I wanted to combine the event stream data with the other data about the customer to get that complete customer view, because we are like kind of developing applications on top of that data. It felt like you already have your data warehouse where you're trying to bring everything together. Why do you want to have a segment like functionality in the cloud? You literally want that functionality on top of your data warehouse.

It's kind of wielded to the first point, but the idea being like you can control the data. You control the flow of the events and it's in your environment. That was the kind of the main driver. Plus, open source felt like – Engineers love open source products. People are building open source, their data stack using open source technologies. It felt like this should also be an open source product that you could just deploy in your environment. The flexibility around it, like if you want internal integrations, you can just add one, because the code is open source. That was kind of the third driver.

[00:10:02] JM: If I understand it correctly, you're saying that the typical workflow with Segment is that I get all my data stored in Segment and I'm periodically going to be doing ETL from a segment into my data warehouse, doing some stuff in my data warehouse. In contrast, you're suggesting an approach where the data goes directly into the data warehouse?

[00:10:29] SM: Yeah. You don't have to do the ETL with Segment. Segment can give your dump into your data warehouse, but that's like once in six hours kind of dump. Think about this architecture, right? There are two pieces to this puzzle. One if just a piping of events. You collect all the events and you send it to all these destinations, right? For that use case, I think Segment

is good enough, like the approach. Other than the privacy and security implications [inaudible 00:10:53] everything outside.

Then comes the second use case of, okay, now you collected all these event, and Segment keeps a copy and Segment has this product called Personas where it lets you slice and dice the data. You can say that like give me all the people who have done event X, but not event Y. You can slice and dice that data and like create audiences and then you can take actions on those audiences, right? You can say that like give me like all the people who came to the check-out page but did not buy. You will send them to Facebook and run some kind of a campaign. That is the Personas product. It gives you a tool to create those audience segments and activate those audience segments.

Now all that is done on top of the event stream data that Segment has. I mean, you send everything to Segment so it knows the user interactions and so on. But in any large company, the event stream is only a part of your data. I mean, you have lot more other data about your customers, which you don't want to send to Segment. I mean, [inaudible 00:11:55] was a good example. We had our billing data about customers and we know how much somebody is paying. How many support tickers they have open. All that data for, various reasons, it was never sent to a third-party tool.

But to create the audience segments that I was talking about. I mean, to create some interesting segments. I'll give you a concrete example. We wanted to find people who are churning and take some action on top, right? To do that, we wanted to combine the event stream data so that we know like what are the kind of activities they are doing in the app, like if their activity level has gone down. But we also wanted to bring in the billing data so that we know there is like a large enterprise customer or is like a small and medium enterprise customer. That was all in the billing records, and we wanted to combine these two data and take some action on top. We wanted to like send that output to a third-party application. [inaudible 00:12:42] in this case to like take some action, like send them a coupon.

That thing can only be done in your data warehouse, because that's where all your data lives. The complete Personas product that Segment built is not so useful unless that runs on top of the data that is in your data warehouse, and that what was the other pain point we're trying to

solve. It felt like you need to bring the Segment-like functionality, whether it's like the data piping, whether it's like the audience creation on top of your data warehouse.

[SPONSOR MESSAGE]

[00:13:17] JM: There are two ways to add analytics to your application, you can build them yourself with basic charts and dashboards using free open source charting libraries, or you can use a comprehensive analytics platform from a partner that you trust. If you've tried to build it yourself, you know that free actually is not so free. There are hidden costs like time, and maintenance, and technical debt, and those hidden costs can really add up.

Check out Logi Analytics. Logi Analytics is developer grade embedded analytics solutions and they make it easy to create branded dashboards and report the scale within your own application. You can stop wasting time piecing together analytics and allow yourself to focus on your core application. You can go to logianalytics.com/sedaily and you can get a demo to see what is possible

[INTERVIEW CONTINUED]

[00:14:31] JM: I want to understand a little bit more what the typical workflow is with regard to Segment. So if like somebody is using Segment – For those people who don't know. Segment is just this gigantically popular customer data infrastructure product that aggregates tons and tons and tons of data all the time and it gets – Data gets saved on Segment's platform. But how do people use segment together with their data engineering stack? Just give me a little bit more. Like let's take a step back and give me a little bit more on the pain points that people typically have with the Segment customer data platform.

[00:15:08] SM: The typical use cases, like you embed the SDKs in your app. You send it to like Segment. From there, your marketing team decides what event goes to what destinations, and they also have the Personas product. As I was describing, you can create audience segments. You can say like, "Not only just send these events. You can also send these users." People who have done certain actions, you can create those audience and then you send those audiences to like, let's say, Facebook, for running a campaign. That's the marketing user case.

Then there is the engineering use case where you take that event stream and maybe you get a dump into your S3 or your data warehouse and you do some analytics on top. You run reports on top of that data or you're running some kind of a recommendation engine and like that and so on. The kind of pain points I've seen for this specific use case. Number one is pricing. We didn't touch upon that. It's price based on MPUs, and like by MPU I mean like monthly unique users. If the unique user economics, that is a lot from product to product. If you are a B2B company, you have a lot few users and they are much more valuable.

If you are a free game or like a publisher, for example. I mean, your per unit economics are much lower. That is one of the main pain points around like Segment that we come across again and again. The other thing is sending everything to the cloud is the other thing I talked about, right? I mean, if the majority of the use case is to get to the data into your own environment, like whether it's S3 or like your own data warehouse, it kind of feels weird to like send everything pipes, everything out to a third-party vendor and then get it back into your VPC, particularly because of all the regulations now. You have to be really careful about what you have built in the event and so on.

On the other hand, if you are just getting inside your VPC straight into, then you don't have to worry about those things. These are I think the two engineering pain points, and the marketing pain point as I was describing was the fact that the Personas audience creation works well when you only care about the event data. The moment you care about other data, it kind of becomes like tricky.

[00:17:23] JM: Right. Okay. The customer data is super sensitive. That's one of the points you mentioned. Having greater control over where the customer data gets shuttled to is pretty desirable. I'm sufficiently compelled by your product statement at this point. For those who aren't following what you're building with RudderStack, is an open source segment alternative. I mean, it's more than that, but that's kind of I think the core in a nutshell use case. If this is an open source tool, you can talk about the architecture. Give an overview for what the RudderStack architecture looks like.

[00:18:06] SM: Yeah. We have this control plane, data plane separation. Think of the control plane as the layer where you go and configure everything, like you set up your sources, your destinations. We have something called transformations. You manage your users. That all happens in the control plane. Then we have this data plane. Through it, all the events flow through. The data plane kind of takes care of like persisting the events. It's sending to the destinations, handling failures, managing event ordering. All that is handled in the data plane. We also have this notion of transformations. Through it, the structure of the event can be changed that is flowing through the data plane. The transformations runs in the data plane.

One example of a transformation is like the – So you get an event in that other format. You want to send it to Google Analytics. You have to convert that into a format that Google Analytics handles. That is also implemented as a transformation, and we also allow users to define their own transformations. So they can structure, modify the event structure [inaudible 00:19:05]. I mean, it's particularly useful if you have like mobile SDKs and you want to change some event after the app has been shipped. All that can be done in the transformation. They're the three main building blocks, the control plane, the data plane and the transformation layer, which runs in the data plane.

[00:19:19] JM: Okay. I think the place to start as we're walking through this is a client SDK. Uber, for example. You get the Uber app. It's on your phone. There is probably some SDK in there that's logging user events and understanding how the user is interacting with the platform to gather data and to make more intelligent offers to that user. Similarly, you have a client SDK, the RudderStack SDK. If I'm building my ridesharing competitor to Uber, I may want to have customer data infrastructure because I'm going to want to track my customers in the same way that Uber does. I have this client SDK and I can write events in my client code. So like in my Uber client, I can say every time a user opens the Uber app, I want to create an event that says user opened app. Then I want to send it to my RudderStack backend so I have a record of this event.

Now, this becomes a really interesting architectural problem because if I'm scripting a ton of events, like if I'm just – A user opens a highly interactive app, they're scrolling, they're clicking, they're swiping, they're closing the app, they're opening the app back up. There are all these events that you could generate. You could continuously send those events to your backend or

you could buffer them. You could have them sit on the client for a little bit and then send them in bursts back to the sever. You've got all these options for how to architect it. Tell me about how you've architected the client SDK.

[00:20:54] SM: Yeah. It's exactly what you said. The client SDK buffers even for some time. They are persisted in a local SQL database. Once it reaches a threshold or a timeout expires, like we send the events once every 10 seconds, and that is configurable. Those events, whatever happened in those 10 seconds or if it exceeds more than 100 events, that's when those events are send to the Rudder backend.

The client SDK handle retries and so on. For whatever reason, if the backend could not be reached, it got an error back. It will keep the event for longer and then retry again and so on. SDKs are pretty straightforward in their error handling.

[00:21:33] JM: What about a high-volume situation like a game? In a game, I can imagine just generating so many events that it would create some interesting problems. Is that a reality?

[00:21:43] SM: I mean. To be honest, that's a fair point. I mean, particularly, if you are on a desktop game where you don't worry about network bandwidth. You could possibly be generating a lot of events. Technically, the same architecture would work. I mean, you are persisting the events in a local store. It's kind of like a circular buffer. Once you hit some threshold, you start pushing it out to the server side. On the mobile games, I mean, generally people don't instrument like gazillion events anywhere to not overload the customer's bandwidth. The SDK side has not been a problem.

Although on the server side, you still need to handle all those events. You have like millions and millions of users and all of them can be hitting the servers. You have to handle all those scale challenges on the server side. Particularly because we are an open source product, I mean, people are deploying it in their environment. It's not a service that we are offering that we can manage to scale up and down. We are to solve some interesting problems around that.

[00:22:41] JM: Okay. On the backend, you have this Rudder control plane and a Rudder data plane. We've already talked through the client SDK. So there are lots of events coming off the

client and they're hitting your backend. You've got this control plane and the data plane. What do the control plane and the data plane do?

[00:22:58] SM: Think of the control plane is the dashboard where you configure the Rudder stack, right? You configure the sources, the destinations, the API keys for all the destinations and the right keys and all that stuff. Let's just say configuration window. No eventful flows through the control plane. Then the actual events flow through the data plane. What the data plane first does is it pulls the config form the control plane. So it knows where all the events have to be sent to. What are their credentials? What are the API keys and so on? It keeps pulling that data.

The data plane is the software layer, which is handling all the events. It gets the event from the client SDK. It immediately persists it in some kind of a buffer. Think of it as like a Kafka kind of stream. Although we don't use Kafka or like learning that aspect, think of it as like some kind of a streaming layer. It persists the event in that streaming layer and then gives an act to the client saying that, "Okay. Now the event has been persisted. You can delete the event."

Then it does the processing, the kind of processing I was talking about. It takes the event and it calls the user transformation to transform the event according to what the user wants. If he wants to remove some fields, or add more fields. There are all kinds of interesting transformations people have written. Like PII scanning is a very common transformation.

It goes through the user transformation. It comes back – With the transformation functions returns another event. Then it calls the destination transformation. Now the event has to be sent to like let's say 10 destinations. It calls the destination transformations to map the event to the structure that [inaudible 00:24:30]. It could be a key value. It could be an XML. All that is handled in the destination transformation, and we have transformation call for like all the [inaudible 00:24:37] destinations we support.

Then it comes back to the data plane, and now the data plane has to send that transform event to the destination. It has to take care of failures. The failures may happen for various reasons. The destination may be down. It may be throttling. All that is handled by the code data plane. That's like the typical flow of an event through the data plane backend.

Now, this is a single node architecture. Now we have also a multi-node architecture. If we want to scale to billions of events, then you have to like – Single node may not be enough. That's why you need a multi-node. I mean, I can get into the details, but that's like slightly more involved than the single node, but that's like the typical simple flow.

[00:25:18] JM: The multi-node situation, does that just happen when I have lots and lots of routs or I have different geos or I have – I mean, tell me more about the multi-node deployment.

[00:25:30] SM: Yeah. There are a couple of reasons you may need multiple Rudder nodes. One is the pure event volume. I mean, typically, our single node can handle from 1,500 to 3,000 events per second. If you need more than that, you set up another node, right? That's like the most common use case. The other use case is high availability. You may want – If your single node goes down, I mean, you may want a spare node running immediately. That's like the other reason.

Multi-geo is also an interesting use case. We don't have a multi-geo architecture yet, but you could imagine that the same thing, like if your users are all over the place, you may want to like deploy a Rudder node in each geo closer to your users and you may have like a cross-geo AWS deployment. You may want to run multiple Rudder in each of those geos. That's another use case.

At a very high-level, the multi-node architecture is very simple. I mean, you have these user's events and you want to just send it to any of those nodes. They are just processing. There is no dependency between events. The only dependency is between events from the same end user. If I'm an end user of Uber and I'm sending all these 10 events, I don't want all the 10 events to go to different nodes, because that might break the ordering of the events. I mean, you have to do some user-base routing, but other than that, it's pretty parallelizable, if that make sense.

[00:26:54] JM: It does. There's a cloud hosted version and then there's the on-prem version. The cloud hosted version, so do you offer as a SaaS that multiple tenants are using, or do you spin up an entire instance of the whole customer data infrastructure stack for each individual customer or each individual company that uses the platform?

[00:27:21] SM: Yeah. We have two versions of that. One is the SaaS kind of version, where we have only one instance of router running. We have that mostly for demo. Anyone who sets up for a free trial and he wants to just – He doesn't want to deploy the whole thing, he can try the free trial version, and that is a single Rudder handling everyone.

Then for our paid customers, we spin up a separate Rudder instance. We are Kubernetes native. So like we have a Kubernetes cluster and spin up a separate set of ports per customer. We create a separate name space per customer, and that's how we handle customer data. They're pretty isolated from each other.

[00:28:00] JM: Got it. The on-prem deployment situation, how do people typically deploy RudderStack on-prem?

[00:28:09] SM: We have all the different kind of deployments we have seen, right? We have people who are deploying Rudder on like on build instances. They'll spin up AWS instances and deploy it out of there. We have also people like just setting up Docker, like EKS or something. What we generally recommend is do Kubernetes, because all the multi-node scale up, scale down, everything, we are doing through Kubernetes operators. If you need one node of Rudder or if you don't want all those like advance multi-node features. Either of them would work, and we have like script for all of them. We have Terraform scripts and Docker images and so on. But like the typical multi-node deployment, we generally recommend Kubernetes.

[00:28:51] JM: Okay. I'd like to talk a little bit more about ETL and data warehousing. Let's revisit this. When people are using RudderStack, are they typically doing ETL out of your backend and into a data warehouse?

[00:29:07] SM: I mean, the backend does not keep any state. There's no ETL. I mean, the events are coming to the backend. We store for some time, because you don't want to send individual events to your data warehouse because of cost and so on. I mean, you'd be batched for half an hour. Then we load all that into a data warehouse. There is no like separate ETL process you need to do. We have the transfer layer. We are transferring the events into your data warehouse.

[00:29:34] JM: Okay. Got it. On the front of usage of data warehouses, do you have any breakdown of customers? Who is using Snowflake, versus Red Shift, versus BigQuery? What are the data warehouse breakdowns for people that are using RudderStack?

[00:29:52] SM: I can talk about – A crossover paid deployment. So people who have interacted with. I would say like it's almost like equally spread between Snowflake and Red Shift and like maybe 10% BigQuery. That's about like people we have kind of interacted with. Think, we are also an open source product. We have like a lot of deployments like where we don't know. We have never interacted with the person. I mean, I don't know what's happening there. But at least for the conversations we have had with customers, I think that's the typical breakdown.

[00:30:23] JM: Right. The fact that people could bring their own data warehouse, that brings up the point of actually integrations, which I forgot to discuss with you a little bit earlier. All these CDI platforms, so you've got Segment and there's mParticle and there are probably a few others, and then RudderStack as well. All of these platforms have so many integrations. Like you want to integrate with Marketo, and MailChimp and all these different things because when an event happens in one of these systems, you oftentimes want to log it to your customer data infrastructure platform. Tell me more about keeping up with all those integrations. That seems tough.

[00:31:01] SM: Yeah. That's a fair point. I mean, I'm sure Segment has a complete infrastructure around that, like they're testing all these infrastructure, like integrations and a big team around that. We are a small company. I think that's something we have to like improve on and so on.

Generally, the APIs don't change too much. I mean, of all the integrations I've worked with, I think Facebook is probably the worst, and they have gotten better now. Earlier, they were like shipping new version every 6 months, 9 months. But otherwise, it's not like they change their APIs too much. As long as you subscribe to their newsletters and so on, you get an update. Yeah, it's an ongoing thing, like keeping with all their integrations.

The good thing though is there always the 80/20 rule. I mean, your top 20 or 30 integrations probably take care of 80% of your customers' usage. As long as you're very good at those top 30, 40, you'd be fine for most of your customers.

The other thing that I've seen is like [inaudible 00:31:59] Segment. They have kind of standardized event structure. I mean, they have said that these are the way events should be generated and so on. A lot of the new age SaaS tools, they kind of conform to the segment structure. They kind of take the same set of events. It's not like – Building an integration for them is actually much easier than like an old school, like SalesForce, or Marketo or something like that.

[00:32:23] JM: There's integrations here in two places as far as I can tell. You have the integrations for the MailChimps and the Marketos and whatnot, and those are basically the ways that people are using Rudder on the frontend. But then you also have to integrate with, I assume, the legacy data warehouses. Do you have to integrate with Teradata or your name your other legacy data warehouse, or do people just figure out their own ways of shuttling data there?

[00:32:50] SM: Yeah. I mean, nobody has asked for Teradata integration, but I'm sure we'll run into that at some point. So far we only support Red Shift and Snowflake and the BitQuery and the cloud versions of it. But I would imagine [inaudible 00:33:06] those would kind of be the same. I mean, everyone supports some way of batch loading files into them. I mean, you dump it. We probably have to just work around that.

[00:33:17] JM: Sure. Fair enough.

[SPONSOR MESSAGE]

[00:33:26] JM: Software Engineering Daily has over 1,000 episodes with lots of interviews with engineers from Google, Facebook, Uber and lots of other engineering companies. We also have interviews with investors. We have interviews about the philosophy of technology and culture and strategy around starting a software business. You can find all of our episodes in the Software Engineering Daily app for iOS and Android. These apps have all of our episodes

sorted and searchable and easy to find in categories with related links and commenting features. You can see our greatest hits, the most popular episodes that have stood the test of time.

If you don't want to hear advertisements, you can become a paid subscriber for $10 per month or $100 per year. Just go to softwareengineeringdaily.com/subscribe. We have put a ton of work into building the apps for Software Engineering Daily. We're creating the best listening experience for our users, and you can check it out today by downloading the Software Engineering Daily app from the iOS or Android app store, and I'd love to get any feedback you have on the apps or the show. You can always email me at jeff@softwareengineeringdaily.com.

Thank you for listening.

[INTERVIEW CONTINUED]

[00:34:51] JM: Tell me about the open source ecosystem. Is it mostly work from engineers at RudderStack itself or are there a lot of contributors from other companies?

[00:35:02] SM: It has been mostly engineering from RudderStack itself. We got a couple of contributions on the integration side, like somebody [inaudible 00:35:09] support and so on, but not beyond that. Again, I mean, we are just a 5, 6-month-old project. Building a community takes time and effort. That's something we have to do a better job, but it's also a matter of time. Hopefully integration is something. I do hope that we'll get some community contribution, and the other is transformation. Let's see how it turns.

[00:35:32] JM: I did see a contribution from the MatterMost CTO. This brought up something interesting, because MatterMost is the open source Slack alternative, and I could imagine it being pretty useful if you want to build an open source tool that bundles in a customer data infrastructure platform. Well, then you could use RudderStack and just have that bundled in.

[00:35:56] SM: Yeah. I mean, I've been trying to reach out to some popular open source projects around integrate RudderStack to collect your users' data, and I've kind of have mixed

responses. Some open source contributors, we don't want to collect any customer data at all. That's what I – It's kind of like yeah, and some people really love it, like it makes total sense.

[00:36:18] JM: You mentioned the community transformations contributions. Explain what a transformation is.

[00:36:26] SM: Think of a transformation as a function. That gets executed as the events are flowing to Rudder, right? They event comes to the data plane. First, a user transformation is called, and inside the user transformation, you can do anything on that event structure. You can, for example, remove fails and like add fields and so on. Then that is passed through the destination transformation. It converts into the destination structure. That's the typical flow.

Now, we have seen like very interesting use cases on the user transformation, the first layer. I mean, one typical use case is PII scanning. Sometimes your events have like PII, like either on purpose or you accidentally embedded, like some engineer forgot to put, like remove PIIs when they're generating the event, like email gets in. But you don't want all those PIIs to flow into your data warehouse and you definitely don't want it to go to like cloud sources, like Google Analytics strictly prohibits PII flowing into them.

Now, you can do that [inaudible 00:37:24] transformation. It's a function and it's on the GitHub repo. It scans, looks for standard PII structures, emails and phone numbers in the whole event. If it finds something, it removes that. That's like one common transformation. Other transformations people have done is like general event structure exchange, right? You have the full name in the event, but your destination expects like first name and last name. You split the name into first name and last name inside your event so that you don't have to change the app.

Another very common transformation is like filtering. A lot of the downstream applications, like in any analytics product, they charge you by the volume of events. But if you are doing analytics, you don't have to do it on the 100% of the data. I mean, if you can sample it down to 10%, your analytics reports would probably be the same. That can be implemented as a transformation. You can like randomly sample events before they are sent to some destination. That way, you can keep all the events in your S3 bucket, but you can also send a fraction of the events to some cloud destination and save a lot on that cost. These are again like some transformations

people have used. I mean, we are creating that repository of like transformations we've kind of seen from our users as well as our internal usage.

[00:38:39] JM: The transformations, they're typically happening on the client side or on the server side?

[00:38:46] SM: It is happening on the Rudder backend. The client send the events to the backend, and that's when the transformation function is called to change the structure of the event. That's why it becomes powerful, because particularly on a mobile world, you ship – Like your app is shipped and your events are fixed. I mean, at that point, like changing any event structure requires you to send another update. But with the transformation, you can do the change on the Rudder backend side. So you don't have to wait for like a new updated version. So simple changes, like if you forgot to split the name into first name, last name, that you can do like [inaudible 00:39:22] on the Rudder backend.

[00:39:24] JM: So are you thinking of yourself mostly as an open source segment or have you strategically diverged in any other way?

[00:39:35] SM: I mean, we think of ourselves as an open source CDI. Yeah, Segmented started the space, but I think overtime we are getting pulled into a very different use case. I mean, we are trying to build a product with ingenious love. Open source is one part of it. But other things like deployment flexibility, right? You can deploy on Kubernetes cluster, inside your VPC. We have like Grafana dashboards around the performance of the events and the whole backend. Things, generally ingenious love, integrations. We have a lot more deeper integrations. For example, we integrated to Kafka, which Segment doesn't have, and so on.

We are trying to build the product for an engineering persona. Again, we are too early in the journey. We'll learn and so on, but just the fact that we are focused on that persona and those use cases, not so much the marketing use cases, I think will take us in a very different journey than what Segment is going through.

[00:40:29] JM: Interesting. Deployment flexibility, what does that look like in practice? What are the deployment pain points and what's the matrix of different deployment options you want to offer people?

[00:40:41] SM: I mean, you want to build heavily on Kubernetes. That's most on the platform layer. But even there, I mean, the fact that you could start with a hosted router. I mean, I don't have the engineering team, so I let Rudder manage the backend. But if tomorrow I want it to take it over, it's literally lifting down the Kubernetes cluster, takedown the namespace and bring it up here. So like in 5 minutes you can move the whole infrastructure from us to them. In your VPC, you have more control. Those kinds of things.

Plus, we do support other deployment options also. I mean, we are an open source and people have changed. We are a [inaudible 00:41:17] thing, like somebody ported it to EKS and then so on. Other kind of Kubernetes environments. All that is possible, again, because of the open source nature. That is what I meant by deployment flexibility. You can deploy in Rudder in whatever environment you are running your stack.

[00:41:33] JM: Got it. What's your perspective on how the landscape of customer data infrastructure is going to change in the next few years?

[00:41:42] SM: Yeah. I mean, that's a good point. Even Segment doesn't use the term CDI anymore. They call themselves CDP, like a customer data platform. All the mParticles of the world, they also call themselves a customer data platform. If we look at the customer data platform space, they're not just these three vendors, but they're like probably under other vendors who call themselves CDP. The Uber idea is still the same. You want to bring every customer interaction into some kind of a single place, like whether it's events, whether it's like the billing data I talked about. Everything that you know about that customer, you bring into a one single place and you take actions based on that. It could be a simple action based on the activity they have done. It could be complicated ML pipelines which predict whether a customer is churning based on all their activations and so on. Then you take some action.

That's where like everybody is trying to go to, the holy grail of customer data platform. It's kind of like an evolution of the CRM. If we think about Salesforce, that was the customer data

platform. You're creating all your customer records in Salesforce, except that nobody was sending event data to Salesforce. Salesforce is just your customer record and maybe you have like a ticketing system and all that stuff. Event data kind of makes it interesting. You don't send that to this kind of old aged CRMs. That's where all the new age CDPs come in. That was my first point.

I think the evolution has been like go from the CRMs of the world to this new age CDPs. But they are still SaaS services. I mean, you send everything to like Segment or mParticle or pretty much all the other CDPs. You send everything to them and you create your customer record in their platform. They keep all the data and they give you all the tooling to create audiences and all the things I was talking about. That's the next phase of like CDP.

What I believe is larger enterprises will want to own this stack. I mean, you don't want to create your entire customer profile on a third-party SaaS solution. You want to like bring it on to your data warehouse. You already have your data warehouse. You're bringing in all the data. If you look at – There were Teradata, and now all the Snowlakes of the world are just like exploding. That's your platform where you want to bring all these data. That's where we want to play. As I was saying earlier, I mean, build the customer profile in our data warehouse and we'll give you the tools to get the data into your data warehouse and create all these audiences. Maybe downstream we'll also build these ML models to create predict churn and so on. We'll give you the tools, but the data stays on your data warehouse. That's I think how the world will evolve around CDP.

[00:44:22] JM: Very interesting. Well, Soumya, what are the difficult engineering problems you're working on right now?

[00:44:28] SM: Two things I would say, like the multi-node is like – I mean, we have one version of the multi-node, which doesn't have autoscaling and so on. Going from a single node to multi-node is pretty hard, and we have a fairly small team. That's like one of the main engineering teams we are working on. Of course, like there are like long-tail of features. I mean, single sign on and all those stuff on the control plane. The other thing is like the integrations that you brought up. I mean, I think an integration is easy. We have kind of made it a framework. But the challenges like how do you like test these integrations in an ongoing fashion. I mean, if you miss

something. That's where some of these destinations don't even give you an error. If you send a random jump to Google Analytics, it will send you 200 okay. Now, how do you measure that that is actually working?

You have to definitely go to the dashboard that it even showed up on the dashboard. All integrations [inaudible 00:45:17] testing our integration is something we haven't done a lot about, but that's something we need to solve.

[00:45:24] JM: Okay. Well, Soumya, thank you for coming on the show. It's been great talking to you about RudderStack.

[00:45:28] SM: Thanks for having me. It was really nice talking to you.

[END OF INTERVIEW]

[00:45:39] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications and it's definitely hard to get engineers excited about maintaining those back-office applications. Companies like a Doordash, and Brex, and Amazon use Retool to build custom internal tools faster.

The idea is that internal tools mostly look the same. They're made out of tables, and dropdowns, and buttons, and text inputs. Retool gives you a drag-and-drop interface so engineers can build these internal UIs in hours, not days, and they can spend more time building features that customers will see. Retool connects to any database and API. For example, if you are pulling data from Postgres, you just write a SQL query. You drag a table on to the canvas.

If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily, and you can even host Retool on-premise if you want to keep it ultra-secure. I've heard a lot of good things about Retool from engineers who I respect. So check it out at retool.com/sedaily.

[END]