EPISODE 1323

[INTRODUCTION]

[00:00:00] JM: Time series data is composed of sequential measurements or events that are tracked, monitored, downsampled and aggregated over time. This could be server metrics, application performance monitoring, network data, sensor data, events, clicks, trades in a market, and many other types of analytical data. The platform, Influx Data, is designed for building and operating time series applications. Influx Data is engineered for growth with enterprise-grade security, ingestion metrics, events and logs in a high-performing time series database and platform analytics for detecting and resolving problems.

In this episode, we talk to Russ Savage, Director of Project Management at Influx Data.

[INTERVIEW]

[00:00:50] JM: Ross, welcome to the show.

[00:00:52] RS: Thanks for having me, Jeff.

[00:00:54] JM: You work at Influx Data. And Influx Data is a category that is broadly known as time series database. My sense is that there's a lot more to the category of time series database than meets the eye. And it feels like just the category of time series database is actually kind of a platform. Do you agree with that statement?

[00:01:18] RS: Yeah. 100%. So, InfluxDB – I think when we first started the company, it was very much focused on the time series database as the center. But I think as the use cases have expanded and as people have built more advanced applications on top of that, I think the database is still at the core and at the center, but you start seeing more and more capabilities needed in order to support some of those advanced use cases. So you start layering on things like background processing, and scheduled tasks. You start layering on visualizations. You start layering on all sorts of other capabilities, that when you add all those pieces together, the sum of

its parts, the platform, is much more valuable to developers as they're building what is essentially time series applications on top of our platform.

[00:02:03] JM: What is a time series application?

[00:02:06] RS: Yeah, great question. So a time series application is usually an analytics application. But it's an application that's centered around time series data. And time series data is essentially any data set with a timestamp that changes over time. So you think about time series applications as analytics applications, you think of it as machine learning and modeling applications. You think of it as just applications on your phone. Anytime you're looking at and you pull up an application on your phone that shows a graph of data over time, that essentially is part of a time series application that could be powered by our platform.

[00:02:43] JM: Do you know the Druid database product? Like the interactive analytics thing? Have you seen that company?

[00:02:49] RS: Yeah, I'm familiar with Druid.

[00:02:50] JM: So is the category of time series database the same as the Druid category? Because when I just think about you want to be slicing and dicing large swaths of time series data on the fly, is Druid in that category, the operational analytics thing?

[00:03:08] RS: I think it's part of it, for sure. So time series data is everywhere. It broadly categorizes into kind of this infrastructure machine metrics and this infrastructure monitoring category. And there's also a huge swath of IoT, industrial IoT hobbyist IoT, sensors out there that is also providing a ton of data. And so I think when you think about those two major use cases, you start looking at where things are optimized for and what tools are right for the job. And I think, from our perspective, we want to satisfy both of those needs. And then we want to build a platform that's flexible enough to satisfy applications that do both the infrastructure monitoring aspect, but also the IoT, the industrial IoT space. And so, yeah, I think 100%.

[00:03:56] JM: Okay, so I've done a few shows recently on this Druid thing. And my understanding is Druid is an advancement on the basic premise of let's make an in-memory

database, which is like great idea. And a lot of people have built cool in-memory-like database structure things. But I think in the Druid world, a lot of the niceties of the Druid database are that you can do this on the fly slicing and dicing kind of processing sort of thing, which doesn't feel like you need to do in every application. There're a lot of applications where you kind of just want the raw time series data, or you want some like simple interpolation, or extrapolation, or I don't know. I'm kind of wondering what the different transactional properties or the different memory or storage volatility properties that you want out of these different classes of database types.

[00:04:50] RS: Yeah, for sure. So I think one of the key differences and what makes the time series database so powerful, it's really, really optimized for writing and querying real time information. So for example, we're talking about – If you think of an individual sensor, or an individual device could have 30, 40 metrics coming off of it. But the people that we're working with and the teams that we're working with, they're deploying tens of thousands, hundreds of thousands of these sensors times 40,000, 50,000, and that is bringing in data multiple times per minute, for sure, and sometimes multiple times per second. So you start seeing these just massive ingestion loads because of all of the data getting multiplied.

And bringing all that data in is one challenge. And you have to use different ways of thinking about indexing and working with that data on the fly. But then also querying that data out, and the idea that being able to access the most recent data quickly, it's not an easy task. And it's something that time series databases really, really excel at. And so it's really that combination of incredibly fast ingestion and querying the real time data that makes the time series database so powerful.

And the notion that, over time, the data that's coming in, it might not be as useful as when you first collected it. And so the idea of being able to shed data that isn't as valuable, that CPU metric from that device, very useful for the first couple minutes, first hours, maybe days or weeks. But over time, you want to aggregate that, such that your storage costs can be kept in checked with the rest of the cost of running the system. So another example of expiring data after a period of time.

[00:06:33] JM: Tell me more about some of the prototypical engineering problems within time series database creation.

[00:06:41] RS: I think the storage and the slicing and the sharding of all of the data coming in is probably the main one. At InfluxDB, we've developed our own storage mechanism for handling all that. We call it TSM. And that's probably the biggest one. I think the other parts are around working with that data and exposing it to applications. And so you need to have a really powerful scripting language, a powerful query engine on the other side that's able to access that data. It's talking really close with the storage. It understands the storage layer, and is able to query that data really quickly.

And so at InfluxDB, we have a language we call Flux that's a really powerful functional query language that you can really do some powerful analytics, and you're doing those analytics as close to the storage layer as possible, which means you're getting the most performance that you can get, which is awesome.

[00:07:32] JM: Why did you create a language?

[00:07:35] RS: So when we first announced that we were building our own language, I think there was a lot of discussion, there's a lot of debate. Quite frankly, there's a lot of discussion internally too. And I think what we came to the realization was, is that as we look at the people that are using our platform, what we started to see is – What we wanted to do is we wanted to bring the compute and the analytics as close to the storage layer as possible. And so in order to do that, what we found is a lot of people who were writing applications against our existing query language, which was a SQL-like language, they were running into limitations where they would essentially dump a large portion of the data out into their application and manipulate it in their own code. And so what we're looking for is how can we start bringing those manipulations closer to the actual platform so that we can speed them up and we can provide kind of mechanisms so that all the users could benefit from these custom functions of this custom code instead of just the individual?

And so when we started to look at the different requirements that we needed in order to do that, it just made sense to take a look at it from a fresh perspective. Obviously, SQL is well

entrenched in the data community, which is why we support InfluxQL as a SQL-like interface into our data. But we also wanted to provide those power users and those people who are really building incredible applications on our platform kind of a language that they could use to really turbocharge their analytics. So that's where we decided to create our own language blocks. And it's really awesome. The team that's behind it has been working kind of nonstop for years on getting that stuff right. And people are really, really having success with it.

[00:09:14] JM: Tell me about standing up this kind of thing into a cloud service.

[00:09:19] RS: Yeah, it's not easy sometimes. So our Influx cloud in terms of how it's architected, it's made up of a bunch of services that run inside Kubernetes control plane. And so we've been slowly rolling out new clusters across the globe over the past couple of years. I don't know. I think we're up to close to 20 regions. But it's interesting, because Kubernetes is really awesome. It's this notion that you're supposed to be able to deploy these services into different Kubernetes instances and have them all work without too many changes. But we found rolling out a multitenant cloud service across the AWS, Azure, and GCP, or Google Cloud services. It's tricky. There're a lot of details and a lot of nuances there. But it's a really awesome and really scalable platform. So that means if we're constantly monitoring all the information coming in from our users, we see a huge spike in query load, or a huge spike in writes. We can scale that, or Kubernetes can scale that automatically. And it creates a really powerful way for us to optimize our cloud service.

[00:10:29] JM: You worked at Elastic for over a year. How does running elastic compare to running InfluxDB?

[00:10:38] RS: I really enjoyed my time at elastic, and I have no negative things to say about them. What I'll say is that the use cases for time series database, at least when I was working at Elasticsearch, which was essentially a search database, they're different. And I think you've seen Elastic – It turns out that when you have a really powerful search capability, you can apply it to a lot of use cases, which is really awesome. You see Elastic start coming more and more into the time series space.

What I think the biggest difference I'd say is the amount of infrastructure and hardware you need to get the job done. And so I don't think this is a shock to anybody. But obviously, Elasticsearch does a lot more processing on the data ingestion side. And so when you're ingesting large portions of data, you need more resources to do that indexing on the fly. So I see that as one of the major differences, is that for the use cases that the time series database solves, I think Influx Data has a lot lighter system requirements for something like that. The stuff that you would need a multimode Elasticsearch cluster to do, you can do with a single instance of InfluxDB open source, or a very small cloud footprint. So that's probably the biggest difference.

[00:11:48] JM: How has the customer set changed as InfluxDB has grown?

[00:11:54] RS: Yeah, it's interesting. The customer set has definitely evolved. Obviously, InfluxDB has evolved as well. So there's no surprise there. But I think we've been around for a few years. And what we've started to see is when we first came out with our InfluxDB database, there weren't a lot of other options out there. You've seen the market explode over the last couple of years. A lot of different options for not only time series database, but also just metrics and monitoring platforms in general. Some of them very purpose built for monitoring very, very specific infrastructure. And then we've also shifted from kind of the on-prem or the posted instance in cloud kind of world into this multitenant shared infrastructure model.

I think what we've seen is users that were, at the time, leveraging a database for a very specific purpose, look for ways to expand out into applications that are dedicated to their purpose, or building their own application dedicated to their purpose. And so I think our customer set has kind of changed a little bit from maybe the operators or the database operators, to potentially the application builders themselves, or the developers that are building the applications. Because you're starting to see more people, more and more companies move towards the developers are also responsible for monitoring and maintaining those services. And so they're looking for technology and tools to do that.

[00:13:20] JM: Do you remember the Michael Stonebreaker thing, the age of one size fits all is over? Is that familiar to you at all?

[00:13:26] RS: I can't say that I've heard of that. Although the phrase that you just uttered, I've certainly heard that phrase before.

[00:13:32] JM: Yeah, so there's this database guy. He's the founder of VoltDB and several other databases. Just serial database entrepreneur/computer scientists person named Michael stonebreaker. I think he first became notorious because he was a Hadoop skeptic. And then he was kind of proven wrong a little bit, but not proven wrong. He had very legitimate skepticism of Hadoop. But then he had this paper called The Age of One Size Fits All is Over. Basically, the idea is you've got multi-model database systems, or multi database, multi model systems. And it really feels like we're there, right? You've basically got a database that's called a time series database. It's like very domain-specific. How domain-specific do we need to get with our databases?

[00:14:15] RS: I've definitely heard that phrase before. I've thought of it as use the right tool for the job. It's not that the tool sets are ever increasing. And so, yeah, you have the option of choosing the right database for the right problem that you're solving. You start seeing – I think the idea of a general data store works up to a certain scale. But once you hit a certain level, you start spending more time on the infrastructure and less time on the actual application that you're building or supporting with that infrastructure because of different concerns, performance, scaling, all those sorts of things.

So, yeah, I mean, at InfluxDB, we obviously believe in specialized databases, specialized platforms for solving problem. I think what we're finding in terms of how specialized you need to be, you need to go, it really depends on the use case and the scale that you're working towards. So for example, if you're working on a platform and – I don't know. You're working at a scale that can easily be solved by a generic database. There's no reason to over optimize before you actually need to, unless you have specific plans or specific goals. I think we're starting to see time series databases become more and more flexible and more and more useful, and actually be able to bring in data from non-time series locations.

One of the things we see customers do a lot is time series data will stream in and then they'll leverage or reference relational databases to enrich that time series data before it's stored, right? And so you get the benefits of bringing that data in, storing all the metadata with the

actual series information so you get fast querying, but you also get the metadata stored in a relational data store that's really, really great for those types of things.

And so in terms of how specialized you need to go, I would say it's kind of a cop-out answer, but it really depends on the use case and kind of the product that you're building.

[00:16:14] JM: What's your modern take on the tension between volatile and non-volatile storage? Like what do you want in a database? When are you tearing to storage? How important is that tearing system? Is it only important to cost? Is it also important to performance? Tell me a little bit about architecture as it pertains to storage systems and the memory hierarchy.

[00:16:36] RS: Yeah. I mean, that question feels more geared towards maybe a college lecture on storage design.

[00:16:40] JM: Is it too boring? Is it too boring? Do you want to talk about developer experience or something?

[00:16:45] RS: No. No. No.

[00:16:45] JM: What do we talk about here? What's your back? What do you like to talk about? I'm open to talk about whatever. It's a deep product.

[00:16:52] RS: I don't think I'm going to break any new ground on how to describe multilayered storage.

[00:16:56] JM: Actually, here's one thing I'll say on that. This is something boot camps don't teach, which is fine. But it's exactly the kind of thing that is useful to learn in a podcast if you're somebody that came from a boot camp.

[00:17:09] RS: Yeah, yeah, it's very true. So in our world, data has – Like more recent data is more valuable in a lot of cases. And so the use cases that we optimize for is being able to really, really quickly access recent data in the platform. And what that means for a storage perspective

is you want to keep that data as hot as possible and as easy to access as possible. Over time, the value of that data degrades. It's less likely that you're going to be querying it. Or when you do query it, you don't need sub-second, sub-millisecond performance. And then you start to look at the equation of storage cost versus access.

And so in our world, the older the data is, it gradually gets moved in. In our world, we call it compacted, but moved to different levels of performance from a storage perspective. So our databases are using SSD storage. Well, it's using RAM for the most recent storage, but SSD for most of its storage, and then it can archive out to slower data stores if it needs to. So, yeah, that's kind of the level that I can get to, is in the time series world, the more recent the data, the more useful it is. And so we make access to that information much faster.

[00:18:20] JM: Where do you feel like the most prominent battles are being fought across the product right now? Like is it about just winning over people on the category of time series? Is it about engineering out the last bugs? Or is it about like building a platform on top of the basic concept of Influx?

[00:18:40] RS: So I think the biggest battles that we're fighting, and I wouldn't call them battles, they're awesome problems to have, is we're looking at – Essentially, we put out this platform. We put up this time series platform. It's a generic platform, and you can solve tons of different use cases on it. And what we're seeing is users are using the platform in ways that we didn't actually think that they would be using them. And they're finding ways to exercise, and I wouldn't use the word exploit, but really kind of hone in and optimize on different parts of the application.

When you're developing, you're building a bunch of different capabilities. A lot of times, those capabilities don't resonate. And so you move on or you optimize somewhere else. And so I think what we're seeing is we're looking at we're really close to our customers that are leveraging our cloud platform. We're seeing use cases. So one of them is this notion of I'm storing my queries in my application. But that's a downside, because anytime I have a bug in one of my queries or I have a problem in one of my analytics, I need to go out and update thousands of clients out there. What I'd like to do is update some Lambda on a server, and then all of my clients automatically see that new information. So the same kind of continuous CI/CD process that we use in cloud, they want to develop for their customers, which makes a ton of sense.

And so we're looking at ways now, which is investing in some of our API's around storing functions and reusing functions and bringing that capability in-house. It kind of goes along the same model of bringing more compute closer to where the data is. So that's just one example of where I think we see users taking some of the stuff that we've built and really running with it and really kind of demanding more where we're investing.

I think the other thing that we're really trying to figure out, I mean, I don't think we've nailed it yet, is the role of our UI in a platform like this. I think our UI serves many different purposes today. It's an analytics tool. It's a resource management tool for the platform. It's an operational tool. It's a lot of different things. I think we're really looking at ways to – It's a developer tool mainly. We're really looking at ways to kind of focus that UI on to developers building on top of our API's and figuring out the best way we can help them develop faster, because that's really what we're trying to do, is we're always trying to help people develop their applications and their capabilities faster and write less code on their side.

[00:21:13] JM: What about in terms of the software development, software engineering challenges? So I know all about how to build web apps. I don't know as much about how to build a time series database. So what would surprise me about the actual software development process of building a time series database?

[00:21:30] RS: Good question. I'm thinking about other projects or other software that I've worked on. I think what's really – I think we may be underestimated a little bit. One of the goals that we wanted with our latest cloud offering was, from an engineering perspective, we really wanted to hone in on the continuous integration, continuous delivery pipeline. Like we wanted changes that engineers we're making to be out into public as fast as humanly possible within the hour once they're checked in in many cases, obviously through various testing and all that sort of stuff.

I think one of the things that I think was harder than we originally planned was getting that pipeline up and running smoothly and making sure that we had the tools and the capabilities needed to understand where the problems were in that whole process, that whole deployment process. Because shipping a new version of an application every minute is non-trivial in a lot of

cases. And this is – It's not one application. It's many different applications that come together to form a unified experience and a platform. So I think that was one thing that was a little tougher. It's pretty unbelievable the amount of engineering that's gone into creating this platform as a service, this shared infrastructure, and I can't even begin to describe all the different layers there. But that, to me, having worked on other web apps, there's a lot of moving pieces when you're building out a database platform, that a lot of times when you're just building UI, you're just building one piece, you don't have to think about it.

[00:23:05] JM: Is time series database ever used as a transactional database? Or is it mostly kind of append only write thing plus read only mostly for aggregations, and roll ups, and kind of things like that? Or is it ever used for transactional data?

[00:23:24] RS: We see very few use cases in our world that rely on transactional data. So people that are building out systems, or building out applications in our system, they're looking for append only. And really, the way to get the right performance that you need for large scale data application is to do that append only. We do see customers coming in and overwriting previous data. But in our world, since a lot of our data is tied to real world events, really, it's tough to go back and rewrite history. And we don't see that use case very often. And if we work with customers and we see them designing a solution that requires them to rewrite history often, we attempt to persuade them that there are other ways, whether it's through like a continual change log that gets updated or something like that. So yeah, in our world, it's very write heavy. We don't do a lot of updates and deletes. And that's by design.

[00:24:20] JM: Again, how should I be thinking of a time series database like InfluxDB, Influx Data in my stack, in my toolbox? What am I doing with this thing? Am I just using it to blindly write time series that seem useful and then later on I'm using it, like I'm sort of querying it as it seems useful? Is it mostly used for metrics these days like a Kubernetes kind of stuff, Prometheus-based logging? I'd really love to know where it shines today and what applications I'm really, really looking to align with this database product.

[00:24:54] RS: Yeah. I think where we really shine is there with customers that many different devices out there, many different kinds of devices are either virtual machines, or actual physical devices, sensor information, all sending data to a central location. I think ingesting all of that

information at scale is one of our strengths. I think, within our platform, once the data comes in, there's this whole notion of turning data into knowledge, right? And so the data comes in, it has very little value, but the more you work with it and the more you clean it, and enhance it, and aggregate it, and run analysis on top of that, it turns into knowledge.

And so one of the things that I think our platform is really good at is finding that knowledge in all of the data that's coming in, and then it actually allows you to develop and present that information to your users or leverage that information yourself to make decisions on the fly. So if you're bringing all this time series data and running it through a model to try to identify abnormalities or predictive alerting, all of that can be done in our platform. And then that information can be queried quickly and presented to users, or yourself, or whoever.

And so back to the standard ingest everything you possibly can as fast as you possibly can, use Flux and the tooling that we provide to extract the knowledge from that information, and then present that information to whoever needs it. So whether it's your end user, or yourself, or whomever.

[00:26:27] JM: have you tried to lever up into the BI layer? Or have you really stayed away from the BI layer and tried to just sit within – Not transactional. I guess, query semantics. It seems like you've really focused on this language, taking this language-based approach seems pretty good relative to like going after the BI layer. Tell me a little bit more about what you've gained traction with and how you think the product is going to deepen its relationship with developers.

[00:26:54] RS: Yeah, the BI layer is really interesting. There's a ton of different solutions out there for doing BI on top of that information. And so I think where we're looking at is being a data source for those tools in many cases. We're really interested in storing and collecting that information, potentially aggregating and analyzing it before it gets to the BI tool. And then making sure that you like any platform out there, you want to make sure that you can connect to the tools that people are actually using. And so we want to provide API's and mechanisms to connect to those BI tools. And so you start seeing – There's different investments you make depending on which tools you want to connect to. And a huge swath of them work with RESTful API's and integrate there. There's a huge category that works with SQL, works with Python, works with other languages. We want to figure out ways to be the data provider for all of those.

We ourselves, I think we stayed away from developing our own BI tool, because the space is so crowded. And I think it's not necessarily where our strengths are today. And then we've purposefully decided to stay at the data provider level in many cases for now.

[00:28:04] JM: As somebody who's worked at several infrastructure companies, you've probably seen the different margin structures that can arise depending on what approach you take as a business. Do you have any general principles, or lessons, or takeaways from seeing various infrastructure companies go to market and how their profit structures have developed?

[00:28:27] RS: My background, I've worked at a couple different open source companies. And so I think I can talk a little bit about some of the open source models that I've seen and how those work with business models. I think, historically, you've got the open source software, and then you sell some sort of support or help on top of it was kind of the original model. That works up to a certain extent. But if you think about it, kind of the goals of your support and the goals of your business aren't really that aligned, right? It's like, yes, we want to help our users, but only enough such that they still need our help in the future.

And so I think you've seen different companies leverage different techniques to avoid that going forward. You've got companies that are really focused on providing key capabilities that are only enterprise-driven and kind of grouping that up into different structures, different tiers, free tier, bronze, silver, gold, that type of solution.

I think one of the things that Elasticsearch has been incredibly successful doing it that way. I think you've also seen the model of creating these hosted instances of this open source tooling and open source software. And for a while, that was kind of our model in a lot of cases. We had a hosted version. It was a hosted version of our enterprise where we were selling by nodes. And I think we've shifted slightly away from hosting our own open source tooling and looking at it from a point of view of like the things that our customers need to run themselves or need to use on their own infrastructure, on their own hardware should always be open source because it's transparency and a level of trust that we want to maintain with our customers.

I think the idea that that software needs to be the same as you run in large scale cloud platforms isn't necessarily always true. So the software that we're running, the API's are incredibly compatible and are the same, but the software behind them is very different depending on where you're running. I think that model, in my opinion, has a lot of strengths where you're still developing and providing a ton of open source value to the community, but you're able to provide value and people are willing to pay for value of operating and running that infrastructure at scale such that they don't need to worry about it.

[00:30:43] JM: What's the future of the company? I mean, one thing I can imagine, for example, is a very heavily event-driven platform. Find certain triggers across a time series, trigger certain infrastructure things off of that. That's a full platform, right? Like the time series automation platform. Is that interesting?

[00:31:02] RS: The future of our platform is 100% moving more towards what they call application performance metrics, APM. Basically, like this event level metrics or the events that are coming through some of these large scale systems, they're massive, and the scale is massive. And so we're investing heavily in next generation storage capability. We're calling it IOx. I think we've got some public talks on that as well. But that foundation is going to allow us to really jump leaps and bounds and be able to ingest an insane amount of events that are flowing through these systems.

I think you're seeing, in the past, it was all about things happening at a regular frequency all the time. So data coming in once a second, data coming in ten times a second every single time. These events that are coming through, they can be three events per second, two events per second. And then suddenly, 100,000 events in a single second, right? And you start seeing this kind of shifts spike incredibly fast. Like during an incident, you obviously want to record a ton more information than you need to on a regular basis, you can go back and kind of see exactly what happened.

And so I think the scales are getting larger. We as a company, we're investing in a ton of R&D to build out a storage layer capable of ingesting and meeting the needs of those systems. And so we look to the future towards the company, we're looking for higher and higher what we call cardinality. But basically, the number of series in your database, workloads that have an

incredibly high number of series, and being able to analyze those at scale. And that's kind of where we're spending a lot of energy is really building that next generation storage layer.

[00:32:49] JM: And what does that mean in practice? Like what is that storage layer? What has to change in the storage layer? What kind of engineering has to take place in the storage layer?

[00:32:59] RS: Yeah, so I can't speak to the specific storage layer details, but what I'll say is there's technology out there, Apache Arrow, that's really, really amazing. It provides a way for you to work with data without copying it in many instances. And so it's probably a really horrible description to anybody who actually knows what Apache Arrow is. But that's not that's not the point. So I think the things that are different are around the set of technologies that you combine together to build a really, really powerful storage layer. They've evolved over the past five, seven years. And so we're leveraging that new technology. We're leveraging some of the new work coming out from Apache Arrow, and we're building a system that is performant enough to handle series in the hundreds of millions, right? So we're really starting to see use cases where creating another database or creating another bucket in our system, that's not going to cut it. You really need to be able to process those large data volumes. And so that's what the storage layer is going to let us do. And it's going to be available in our open source as well. So our goal is that the storage community as a whole can benefit from this work too.

[00:34:13] JM: Let's pull back a little bit. I want to know a little bit about your history. Because you've done a lot of various applications work. Actually, your background fascinates me. Can you tell me? What was the responsibility of a marketing engineer at Box in July 2013? Serious question. By the way, my company, Software Engineering Daily, we're a marketing company. So that's what we are. So marketing technology fascinates me.

[00:34:46] RS: Yeah. Well, it fascinates me too. So yeah, how did how did I end up being a marketing engineer? So I had a background in engineering, and I switched over to being a product manager at a previous company, orbitz.com, which I think that was Sort of Expedia. But at the time, we were working on a project of working with Google AdWords for keyword bidding on travel websites. So basically figuring out a way to keyword target millions of different destinations and destination combinations for search. And we're building a ton of technology there. And so through that process, I learned an amazing amount about how Google AdWords

worked, how the paid search world worked, how tracking in the marketing world worked such that you could get all the tracking information correct. How campaigns were set up, and structured, and tracked, and analyzed and all that sort of stuff?

And so what that kind of led me to is this notion, which I'm still very much in favor of, is that I think the marketing is incredibly metrics-driven. And the technology that's in the marketing space is some incredibly advanced in technology. And I think it's underutilized by a lot of marketing companies out there. And so my role at Box as a marketing engineer was to work with all the individual marketing teams. Ensure that all of their marketing efforts were kind of optimized and tracked at a level that let us actually see exactly where our money was going and optimize for ROI. And it's kind of the full time role, because there's just so much technology baked into a lot of the efforts out there, whether it's through some of the automated email platforms that are being set up. Some of the running campaign tracking from all the different sources. And also doing that analysis on the information, right? Bringing all that information together and presenting a view that makes sense and is accurate, in my opinion, is a really powerful position.

And so that was kind of my role there, was making sure that all of that stuff was working behind the scenes technically. And so I will 100% tell you that I am not a marketer. But if you pair me with a marketer, in my opinion, we could be much more successful in what we were trying to do because we were able to kind of – They were able to come at it from a marketing perspective. I was able to come at it from, “Well, this is the underlying technology you need to set up in order to meet that goal or run that campaign.” And so it was a really awesome combination. And I'm still incredibly fascinated with marketing technology, and SEO optimization, and all that sort of stuff. It's like it's a side passion of mine.

[00:37:29] JM: Hey, maybe we can make it your main passion. There's got to be some marketing tools that you could build on top of a time series database.

[00:37:35] RS: Oh, yeah, for sure. The amount of – Actually one of our customers, Wayfair, leverages our platform to look at data of customers that are browsing their website.

[00:37:43] JM: Really? Wait. Like clickstream or mousestream sort of data?

[00:37:49] RS: Yeah, yeah. It's all time series data. It's all flows in there. They're leveraging it for analyzing that information. So yeah, there's a ton of marketing applications especially as you look at people's behavior as they move through different products, different websites, and different actions that they take on the web, right? And then correlating all the information together. I think it still fascinates me the real time ad bidding space and how that system is set up and works at such scale is really, really unbelievable.

[00:38:18] JM: Do you notice how it seems like every year, the infrastructure companies get a little bit closer to being able to build applications on top of that infrastructure? I mean, at least I would say the better infrastructure companies always go higher and higher level. And it's just sort of interesting, because it's kind of – It seems like if you are anything like an infrastructure company, you're going to become a platform company. And if you're a platform company, you can become an applications company.

[00:38:45] RS: Yeah, it's really kind of an evolution. I think it speaks to the number of resources and the scale that you need in order to – And the expertise that you need to be successful at each level. I think when you start getting towards the application side, you need a ton of domain experts in those particular areas in order to develop applications that are worth anything. And so it takes a ton of resources and a ton of time. And so I think you see companies as they grow and as they scale, they start looking at those different areas as a possibility. If you start trying to do that too early, in my opinion, you really start stretching your resources thin. You start building solutions that are incomplete or not useful to any of the end users that you're trying to target. And so, yeah, I definitely think of it as a process or as a lifecycle. And it's just, in my opinion, has to do a ton with resourcing and expertise at every level.

[00:39:40] JM: What do you think Twilio has become such a platform? Like Twilio was one of these companies that looked like just infrastructure, then they've just become a behemoth.

[00:39:51] RS: Yeah, Twilio is the company that all of their companies look up to and want to be like. They hit such an incredible pain point that they were able to capitalize. And I myself, I'm a Twilio user, and I've written applications on top of Twilio API's. It's an amazing experience. I can't say bad thing about them.

I think what's really interesting is they were able to break into an area that was kind of like individual developers would never even be able to consider building things in that space. Twilio made it accessible. They made a technology that's ubiquitous. They made that technology accessible to every person. And as a result, I mean, they've been incredibly successful. And they've seen – I think, in general, we've just seen a ton of innovation come out of – It's measurable how much innovation has come out of just unlocking that world of interfacing with a device that everybody has, the telephone?

Yeah, it's an incredible company. I think where we look at them very closely is how they were able to develop an API that's incredibly easy to use, that's incredibly simple, but powerful. It's able to solve use cases very, very well. And when you're working with it, you're just thinking to yourself the entire time, “Oh, that makes sense. Oh, that makes sense. Oh, that makes sense. And you can kind of guess in the right direction as long as you understand what's going on. So I think their API design is top notch. I mean, it's something we look to emulate, to be honest, trying to make API's that are simple and powerful and easy to use.

[00:41:24] JM: Alright. Well, as we begin to wrap up, give me a little bit more about – Let's say, I'm a developer. Yeah, it sounds interesting to be using a time series database for various things. Help me calcify in my brain, if I'm a developer, what are the use cases where I should be using a time series database? And how I should be evaluating my time series database options?

[00:41:47] RS: Yeah. So what I would say is the way to know if you need a time series database sounds cliche, but if you have a time series data set that you want to analyze, if you are building an application and you realize that one of the core fundamental components of your application is data that changes over time, then I think you are a good candidate for a time series database. What I would think about when I'm evaluating different solutions or different options, I would look at the whole purpose of going towards a specialized platform or a specialized database is you need to write less infrastructure code, less code to get something to work and more code that provides business value to your company. And so, in my opinion, when you're evaluating different platforms, you should look for what you need to do out of that platform. How easy it is to accomplish that? And how much code you have to write in order to

meet that goal. And look for the platforms that reduce that number, because writing less code is a great thing. So you obviously look at the performance, and the capabilities, and the different interfaces.

I would also look at the communities. I think a lot of these software companies, a lot of the platforms, you have varying sizes of community where you can get help and talk to other people that are leveraging those tools. And so I think that's another important aspect to consider, is if you do run into problems, how can you find help quickly? Where can you get information quickly? Community is really powerful for that.

[00:43:20] JM: Cool. Well, anything else you want to add about Influx or anything else that you're working on?

[00:43:26] RS: Well, I'm clearly a super fan. But no –

[00:43:28] JM: You're a super fan of the database or the podcast?

[00:43:31] RS: Yeah, super fan of the database. I think the time series space is really awesome. I am really excited to see the new stuff that gets developed all the time. Whenever I talk to customers and see what they're working on, I'm in constant awe, just the stuff that's being developed out there. And it makes me get out of bed in the morning. So I really enjoy it. Yeah, I think that's about it.

[00:43:50] JM: Cool, man. Well, thanks for coming on the show.

[00:43:53] RS: Yeah, I really appreciate you having me, Jeff.

[END]