Transcript provided as a free resource by 

Data Mesh Radio Episode #69: Getting Data Sharing Right at Netflix Scale
Interview with Justin Cunningham
Listen (link)

Transcript provided as a free community resource by Starburst. To check out more Starburst-compiled resources about data mesh, please check here: https://www.starburst.io/info/data-mesh-resource-center?utm_source=DataMeshRadio

Scott Hirleman
A written transcript of this episode is provided by Starburst. For more information, you can see the show notes.

Adrian Estala- Starburst
Welcome to Data Mesh Radio, with your host, Scott Hirleman, sponsored by Starburst. This is Adrian Estala, VP of Data Mesh Consulting Services at Starburst and host of Data Mesh TV. Starburst is the leading sponsor for Trino, the open source project, and Zhamak's Data Mesh book, *Delivering Data Driven Value At Scale*. To claim your free book, head over to starburst.io.

Scott Hirleman
Data Mesh Radio is provided as a free community resource from DataStax. Welcome to Data Mesh Radio, produced and hosted by Scott Hirleman, the founder of the Data Mesh Learning Community. Data Mesh Radio is a vendor independent resource for learning more about data mesh. Let's jump in.

Bottom line upfront, what are you going to hear about and learn about in this episode? I interviewed Justin Cunningham, who worked as a Tech Lead and Data Architect on data platforms at Netflix, Yelp, and Atlassian over the last eight and a half years. In that time, Justin was involved in initiatives to push data ownership to developers and domains.

An interesting point, one I'm not sure is universal, but at least it's useful to consider, is Justin saw a lot of success at Yelp focusing on data availability. Getting data to a place it could be found and played with, experimented with, was a bigger driver for success than focusing initially on data quality. Once people discovered what data was available and how they might use it, the organization was able to work towards getting that data to an acceptable quality level. We kind of talked about this in one of the Mesh Musings about the concept of a speculative data product.

Another point Justin made was figure out which you want to optimize for in general, getting things right up front or testing, changing, iterating. He believes in optimizing

for that change, create an adaptive process and optimize for learning, and you'll get to where you want to go in the end, in a much better and easier fashion. Keep it simple and focus on value delivery, it will set up more tractable events.

At Yelp they were trying to ETL a huge amount of data in their data warehouse to build reports for the C-Suite, but they were never really going to get enough data ingested to really meet their goals. It was taking them two weeks to create each new set of ETLs, and that was just creation, not accounting for the maintenance. It was looking like they'd need something like 5x the number of people to really hit what they were trying to do. What Justin found the most useful at Yelp was to focus on getting as much "usable data" in an automated way. They achieved this initially through the data mesh antipattern of copying directly from the underlying operational data stores and then building business logic on top of it. But that data getting into the hands of the data team meant there could be an initial value assessment. Once they proved out there could be value in the data, the conversation with the domains, the developers, the real people who knew the data best, was much easier to get them to care about providing clean and reliable data.

Justin mentioned the same thing Wannes Rosiers mentioned in his episode, there are operational and analytical workloads, but there should absolutely not be that separation when it comes to data.There isn't operational and analytical data, data can be used for whatever purpose is good for it to use for. Data from operational systems is useful for analytics and vice versa. One thing that really helped developers understand how to share their data was thinking of datasets as being similar to public APIs, so when you do go to them and to talk about that, about sharing their data, if you kind of connected the API concept, that helped a lot in Justin's experience.

At Netflix, there were just too many bespoke datasets. It made it very hard to manage quality. What they found that worked was a data certification program for datasets, creating tooling to prove a dataset was complete and accurate. That, and upping the amount of focus on dataset reuse, significantly help them to combat that data sprawl.

Back to data accessibility and data availability versus quality, for Justin, he believes data analysts and data scientists initially care far more about getting data access, as you can work to improve the data quality later, especially if there is a clear owner. I discuss this again in a Mesh Musing about these speculative data products, but a key hack for them was being able to mark data as low quality. This was something I talked about, if you say all data that's on the data mesh is of high quality, people automatically will move to start trusting it. So how do you make it so that people can understand that they've gotta do their homework? Is this data in a format that's

really of use in production or not?

On driving buy in for data producing teams, Justin again talked to proving there was value in data before the producers were bought in. Asking them to serve their data upfront without a clear specific use case was very tough. The return on investment or ROI was very squishy. Why would they wanna do this? So they got out low quality data initially and then came back to producing teams to up quality and reliability once they proved certain data was valuable. This is somewhat similar to the emerging data mesh pattern of creating your data products for a consumer focused use case. It might be a sourcelined data product, but it should still be initially serving a specific purpose with a targeted outcome, and then the data product can grow from there.

To sum up one of Justin's points he touched on repeatedly, he recommends to create a pool of low effort data which will inherently have low quality. Use that for initial research into what might be useful. Focus on maximizing accessibility. You can have governance and use things like joint restrictions or give consumers an ability to self certify that they are using the data responsibly. Once you get to the use cases, then you can go for the data mesh quality data products.

Justin also shared his thoughts on how the way we do data lineage is broken. We should look to do data lineage declaratively instead of just as a reference. That should flow through both the schema registry and the data catalog, what is the data movement supposed to be. This would enable us to much more easily test data flows and alert downstream users of upcoming changes that might break what they're doing. So with that, let's go ahead and get to the episode. With that bottom line upfront done, let's go ahead and jump into this interview.

So super, super excited for this episode today. I've got the world famous Justin Cunningham, who has been doing a lot of stuff, very interesting things in the data space for a while, was at Netflix with their naming collision type of thing called Data Mesh as well, that just has a similar goal, but different approach. We'll get into a little bit of stuff like that, I'm sure we'll kinda poke on that, but what we were really looking at talking about today was organizational challenges relative to data mesh, because a lot of people are running into this. One of the three big questions that I get is, "How do we get these pesky developers to just give us their freaking data? We want them to do what we want them to do, so make them do it. How do we make them do it?"

That organizational challenge issue is something that a lot of people have talked about, and there's just not as much of really good resources out there about frameworks on how to do that. And then we were also kind of just in the pretalk talking about lineage and things like that, and I think Justin has a really interesting

approach to that, so hopefully we'll get into a little bit around how we might be able to think about lineage as well. I know Zhamak has said maybe we don't even need to put lineage in, and it's like, well, once we get to super super duper trustable data products, maybe we don't. Maybe it isn't as needed way down the line. But until then, people aren't gonna trust unless they've got the lineage.

So with that kinda upfront about what we're gonna talk about, Justin, if you don't mind, if you could give people a bit of an introduction to yourself and your background, and then we'll kinda jump into the different topics at hand.

Justin Cunningham
Sure. I'm Justin Cunningham, I've been working in the data space probably for about 10 to 15 years now, in both my own startups early on and then in some big companies, most notably, Yelp and Netflix. And in both Yelp and Netflix, I ended up working on projects that are pretty similar in concept, or at least in terms of goals to data mesh. The Netflix Data Mesh project, as Scott mentioned, which is kind of an unfortunate naming collision. And then before that, a collection of projects at Yelp called the Data Pipeline, which is really kind of like the real time streaming infrastructure to be able to move data around the business, with the aim of really pushing the ownership of data out to developers and product teams. So I've been working in the space practically for a long time.

Scott Hirleman
Yeah, and that ownership on the developer side is, whether that's via technology or just an organizational structure, I think it's a major, major challenge. So why don't we start with what you saw that... Were you brought in originally at Yelp or Netflix to work on that? Or was that something that you came in and then they were like, "We need to do this."? How did that start to evolve, so that way we can talk about how people might take your learnings, and getting started or down the road as to what challenges came up of all those things.

Justin Cunningham
Yeah, great question. And it's pretty different in both places, so I'll start with Yelp's since that one was earlier. At Yelp, I was working on a team called Business Analytics and Metrics, the BAM team. And what the BAM team was basically trying to do, it was their first attempt at building out a data engineering team, they were trying to ETL a bunch of data in the business into a centralized data warehouse so that a analysis team could get a better understanding from a data perspective of how the business was operating. And that analysis team was largely building reports and stuff like that for the C-Suite, so that they could drive into the business. I got to looking at it, and the challenge that I saw was that with the number of engineers that we had, we were never actually going to finish the project. That is, we weren't gonna have

enough data ETLed in, in a usable state so that we would actually be able to succeed in our goals. We were going about this in the normal data engineering way so we're basically taking a collection of all the data in the organization and then trying to prioritize it and getting through a few datasets. It took maybe two to four weeks to build out a new set of ETLs, and then you had ongoing maintenance from there.

So the overall organizational size that we would have needed to actually be able to succeed would have been likely five times or more than the number of people that we had, and so I started looking for a computational or more infrastructural approach to be able to get the data. Which kind of led me into looking at something like data mesh, where we could move ownership out to product teams and focus more on getting the data wholesale, focusing more on ubiquity rather than on quality, at least initially, so that we could get the data first and then focus on improving the data quality later on. I worked on that for about seven years, so we got to go full circle and really toward the end, focus much more on the data quality side of things.

At Netflix, when I started the project there, I was actually looking at it and I didn't wanna do the same set of work again, so I was actually trying to avoid it initially. As it turned out, there was already a bunch of work underway. There was a project called Delta, which was pooling data in the studio space and trying to build out a centralized data warehousing collection, building a bunch of infrastructure, and there were infrastructural challenges associated with that because of the way that it was designed. So I effectively took a bunch of the learnings from what we'd built at Yelp and kinda pulled that into the Netflix Data Mesh product, and then changed the scope of it a bit, expanded it and really adapted it so that it would work well in the organization. And then obviously Netflix has got world class engineers, so it presented a unique opportunity to build something infrastructural that would end up working really well at large scale and taking some of the concepts that I developed earlier and refining them to be a really a great fit, I think, for the business as a whole.

Scott Hirleman
I wanna go back to one thing you said about the Yelp thing, which I thought was really interesting, was started by caring more about the data availability than the quality initially, and I thought that was really interesting. Could we talk about what you mean by quality there, because what I'm finding within data mesh of the people who are kind of succeeding early is that they're caring about the quality as in the trustability and that they understand what this data is, that it's clean data, but it's not that it's the most high value data, it's not that it's the most processed data, it's not this thing where you go, "This tells us all of the information," versus like, "Hey, let's get in the mode of sharing our data in an appropriate way that people can actually

understand what it is and how much they can trust it," and that you can have those kind of differing levels of trust and that's fine. But like, is that what you were aiming for? Or was it even just like, "We just need data so that if we can get to it, we can clean it when we need to clean it."?

Justin Cunningham
More the latter, actually initially. And I'll say that with some caveats. So the thing that we were kind of competing against, which I think is helpful to understand, is really engineering teams basically taking dumps of their databases, or actually in this case, the analysis team sometime connecting directly to replicas inquiring the underlying data, and then because of the data and not actually matching, in the sense that there's business logic applied to data from the services and from the monoliths, that sort of thing, the data doesn't actually in the database necessarily match exactly what you'd want from an analysis perspective.

So they were able to get to some of it that way, so what we started to do was actually pull the raw data out of the databases and then we build some programmatic infrastructure to be able to look at some of the business logic that existed in the monolith. So we had a whole modeling layer that encoded a bunch of business logic, so we could tell for example, if a field was a UUID versus some other kind of field that would have a comparable type. We could also do things like automatic transformation for enums so that they were understandable. So instead of it being like “field value 3”, it was like, "this is this type of business" instead, and we were able to do a lot of those transformations in an automated way. So the goal that we had was really around getting 100% of the data across services and across the monolith, with completeness and with understandability, but not necessarily focusing a ton on the usability side of things.

I would differentiate... When I think about a data product, I think about something where like you've got somebody sitting down and purposefully designing and trying to make something that's consumable, whereas we were looking at what is the most usable thing that we can create with an automated system, so without anybody having to manually do anything. With one additional caveat, and that is, eventually we would certainly like to be able to go back in and improve that data quality over time and actually make it much more usable, but I think the trick there is to have teams care. Because fundamentally, the initial problem that you run into with this kind of thing is that, that many product teams just fundamentally don't care that much about the data, they don't see it as valuable. It's hard to prioritize it against feature development, effectively.

The way that we found that we could combat that is basically by showing if there's actually value in that data, and then when you come back with a concrete ask to a

team and say, "Hey, we've developed this analysis, this data is not really important for driving the business," it's much easier to get them to prioritize making a highly usable version of that dataset, especially if it's useful for them as well.

Scott Hirleman
Yeah, this is that chicken and egg scenario of, "We can't tell you how valuable your data is until you get it into a state where we can use it somewhat," and this is where I think the folks at NAV in their episode were talking about they're literally using cake to entice teams to share, they ship them a cake. If somebody publishes the data product under the data mesh, they get a literal cake. In which people are being like, "What does cake mean?" It's like, "No, it's an actual physical... "

Justin Cunningham
That makes a lot of sense to me. I've got a great example of finding data value in unexpected places. One of the projects that we've worked on that ended up being super valuable in the data space was basically one that was extracting information about which kind of camera was used to take a picture at Yelp. And this might seem like it would be really low value data. I think if you were actually designing purposefully a data product, you would probably say, "This is a vanity metric piece of information, why would we prioritize this?" and maybe even strip it out. But this ended up being a piece of data that was just incredibly valuable. The reason is and the insight here is, a data scientist ended up with an hypothesis that if you looked for pictures that had been taken with really high quality cameras like DSLRs, the kind of person that's gonna go to a restaurant and take a picture of food with a DSLR camera is probably not a bad photographer generally, and that was kind of the bet. And what they found actually is that if they trained an ML model against photos taken with the DSLR cameras and assumed that those were high quality photos, those photos ended up actually being in practice, like the highest quality photos on the site, and you could use that as a training set to kind of seed out a filter that would then select for high quality photos. It made actually a pretty significant improvement in the metrics across the business as those higher quality images were displayed more prominently, and it became something that became almost a core metric. So you could say like if you've got a business and it doesn't have any high quality photos, like what happens if you start adding high quality photos to that business? For example, how does that change your rank and search, how does that change your engagement on the page, and so on?

And then that kind of thing is where I kind of push toward ubiquity initially actually being super important, because once you've got that insight that, hey, we can use this seemingly irrelevant piece of data to drive a bunch of value for the business, getting teams to prioritize and cleaning up that data and making it much higher quality, 'cause initially it's obviously not great quality, it becomes much easier after

you've got that initial proof. And that's the kind of thing in the data space that I think is actually very, very common. And that's why I tend to still lean more toward the ubiquity side of things early on, and actually on an ongoing basis, and then pushing on the quality where it's been clearly defined.

Scott Hirleman
Yeah, and kind of what you're talking about it though is very much the ethos of data mesh, in that it is about sharing data that could be used for multiple purposes. Instead of what we've seen historically in the data space is everything is custom built, custom built, custom built, and that it's very overly fit to the specific purpose instead of reusable. And so thinking about that reusability is really crucial, but it's interesting that even the kind of quality metrics where I think you can set yourself up for a lot of pain and challenges if you give the domains a little too much leeway and you don't have to give us good quality data. They'll be like, "Well, we never have to." But yeah, I think that's an interesting insight there.

Justin Cunningham
Actually, can we double click in on what you've just said for a second, because I think that's also another really interesting thing that we did at Yelp, also at Netflix for that matter, and that is really trying to blend operational and analytical. Because those two have been blending, I think from a technical perspective for some time. What I mean by that is there's analytical uses of operational data and there's operational uses of the analytical data. So for example, you could develop some metric or an ML model that it gets used in your product, it's basically an operational use of analytical data, and you can go the other way in the same way.

What we ended up finding that worked really well is basically building out a bunch of infrastructure where if a team were to create something like a dataset in a streaming system, and that everything that we did with streaming for this reason, they could use that data operationally and then we could make of it analytically. So any piece of data that was ever used in a streaming setting, we basically took advantage of that data and used it for analytical purposes as well. But it was really valuable for teams because they could create a dataset that they intended to use for something like search indexing, and they would create a relatively comprehensive dataset, it's used inside of the products, it's audited, they care a lot about the data quality there because it's fundamentally powering like a core experience.

And if we then are able to take that data and put it in the data warehouse, we've got then a dataset that has multiple uses that teams are kind of naturally taking advantage of. The other nice thing about that is that we started to think about datasets in that way, as like a public API, and we started trying to treat them as closely as we could to the way that we are treating public APIs. The reason for that is

engineering teams, especially product or feature engineering teams, already build and maintain public APIs, so they understand the hygiene around that, like, "I can't make a breaking change here because there's people relying on it. I need to understand who my consumers are, I need to understand what's gonna happen downstream if I change this. I wanna have an understanding of all the systems this is going to. I wanna understand what it's powering. I wanna have insight into the fields that are being used," and so on.

So all of a sudden, because we're using that data operationally, it's starting to meet those product engineering teams where they're at, and they're starting to care a lot more about building those high quality datasets. I saw the same thing at Netflix actually, in a slightly different context, where we had a lot of teams that were building bespoke datasets, where they were building a data feed for different teams around the organization that needed to be able to do something, and then started naturally wanting to condense the data feeds.Because they started to realize that they had five or six different data feeds basically serving the same dataset, and if they were able to reduce into a single data feed that wasn't special built, purpose built for one individual use case, they could maintain a single feed that was really high quality, again, like the public API. And then from an analytics perspective, if we're able to grab those data feeds and automatically warehouse them, we can kind of take advantage of that work that the product team is already doing, and then build out a bunch of tooling to do things like certification of those datasets so that they're more broadly exposed from a discovery perspective, and apply a bunch of data quality tooling around understanding that that data is actually complete and accurate, because it benefits the engineering team to do that as well.

So it becomes very much in honor of that melding of analytical versus operational into one, to really change the culture around data so that it's something the product teams want to own instead of something that you're trying to kind of push on them.

Scott Hirleman
Yeah, and I think within data mesh, as Zhamak talked about it, people kind of make this mistake, and this might just be my own interpretation of, that they think that there is operational data and analytical data, versus there is dataset up for operational work and for analytical work, right? So an operational system absolutely should be consuming analytical data. Wannes Rosiers in his episode, which was like one of the very, very early episodes, talked about he's got three different types of data products, which is kind of this primary source data product that's mostly the rawish type data. And then something that's much more formed around a consumer driven use case, whether that's consumer aligned, resources aligned, it doesn't really matter in the data mesh definition, but then he's got data application data products.

And data applications are literally, there's something that lives on the mesh, just in case other people want to consume from them, but they are designed to power an operational use case. It's not that it's an operational workload, it's not a transactional thing, it's not whatever, but it's like these operational systems need analytical data. And so there wasn't a pipeline, it wasn't something that you're having way too much sprawl from, very bespoke. It was like, we're maintaining something in a productised way that these operational systems consume from. And they have SLAs and they understand it, and they know what's going to be there. So we have a product for those operational systems to be able to consume from.

And so I think that's exactly what you're talking about. Within data mesh, you should be thinking about your data flowing in a full circle. Everything should be feeding into each other when it makes sense, when it's data on the outside. Data on the inside, you don't have to share absolutely everything that you've got in your operational system, 'cause some of it really isn't useful. You may wanna put out that camera data. That might not have made it into an actual data product initially, versus it might have been somebody who puts out a speculative data product of like, "Hey, we've got some speculative datasets. They don't meet our quality threshold to be a data product, but is this useful to anybody? Does anybody wanna play around with this? Is this something where you have that kind of ability to data spelunk around and just poke at things and go, "Is this useful? Is this not?" Like those correlation causation type things.

And I totally get it, but that you can be like, "Well, maybe the high end restaurants have people that go to them with DSLRs," and then you could go to a restaurant and go, that's a partner and you say, "Hey, we wanna test this out, can you get somebody that's just gonna come in, pay 'em 20 bucks to come and take high quality photos of the food. Let's get 50 restaurants in two different cities to do that." And they do, and all of a sudden the engagement just shoots up. It's like, "Okay, we've got a packaging that we can go to these customers with." Your partners or whatever you wanna call the restaurants in the Yelp business. That you've got that data to back it up. But that you don't have to start from every bit of data has to be shared in the highest of quality. That's speculative of like people wanna be able to test this out and just play around with it and see, "Is this something that we should invest in?" and then there's a pathway to getting that higher quality.

Justin Cunningham
Yeah, absolutely. In my experience, very much the problem that I hear data engineers... Well, really data engineers to a lesser extent, but data analysts and data scientists complaining about is a lack of access more than lack of quality, at least initially. Because once there's that insight, it becomes something that is much easier to poke prod on in terms of improving the quality overall, and also making sure that

it's supportable, getting an SLA set.

Interestingly, one of the things that you mentioned, the certification of datasets I think is kind of interesting, because what we actually found was that teams were much more inclined to publish data and maybe to advertise it even, if they could mark some of that data as being of low quality, actually of beta quality or something like that. So the sell on it wasn't actually being able to say, "This is the high quality dataset, and this is a super certified dataset." It was being able to say, "No, this is the experimental alpha version." If you use this it is subject to change, and it's mostly just valuable from, like you said, a spelunking perspective. That really resonated with teams because that was actually a problem that they had, was finding that somebody had discovered their data and then started using it in production, and then all of a sudden now it's something that I've got to support versus having clear expectations set up for it.

Scott Hirleman
I heard this story, and I won't name names, but a very, very, very large company, somebody was kind of poking around and in their warehouse or their lake they found this dataset and they used it for training for something really, really crucial to the company. This is a many, billion dollar plus type of product that they're using it for, and the person who created it had no idea, and they were like, "That data is like six months old." It was high quality enough when you made it, that it's great. It's like, okay, but have you created the new sets of pipelines to make sure that it's all... Have you created the... Or if you're thinking in the data mesh sense of have you created the product to make sure that this actually has a firm backing? And how did you certify that it was high quality when you didn't talk to the person who created it?

There's so much demand out there, but I actually did an episode about this, or part of a Mesh Musings, about the idea of a speculative product. You think about MVP, minimum viable product, there should be a stage before that to talk about, "Here's what I'm thinking about bringing out. Is this gonna be of any use? Does anybody wanna play with this?" Because if I'm gonna spend the time to bring it even up to the quality of MVP, and we wanna get to a place where we can make it so that it's pretty easy to put out a data product in five days, two weeks, whatever, of work, but we're not there in a lot of organizations yet, especially initially.

And so figuring out that kind of thing of not committing to, "I'm going to continually create this," like let's actually figure out who might wanna use this and that you can say, "Okay, if I've got a consumer coming to me and saying I want data, okay, then we're gonna design a data product that has reusability, that isn't specifically bespoke to this, but that fits your use case, and then we're gonna make it extensible and all that." But that there's also this like, is this data on the inside or is this data on the

outside? With the data on the threshold, you wanna make it so that teams can feel like they can share that, but how did you find that? It sounds like it worked well. Did you have any mechanisms that you think that worked? Or was it just being able to be like, "Hey, we're gonna make it so you can kind of put out crap data and it's not that big of a deal."? Because then people can figure out if they want this more.

Justin Cunningham
Well, I think it becomes a question of who's responsible for putting the data out. What we weren't doing concretely is going to teams and saying, "Hey, you guys need to do this work to publish this data." 'Cause it's really hard to get them motivated to do it. And especially to do anything that's not crap data, basically, by asking them to do it when there's not a clear justification on the other side, which is super hard if it's basically for research.

And this is one of the big organizational challenges, is that you basically have a research function on one hand versus a production function on the other. The research function is effectively saying, "We know that there's gonna be some value in some of this data. We don't know which. And there's potentially work to unlock it." The production function is weighing that against, "Well, we are talking to customers and our customers are saying they want X and they're willing to pay Y for it, and we know it'll cost Z to build." So it's much more maybe solidified in terms of what the value delivery is, and that's really what engineering organizations are set up against.

So for that reason, it's always hard to have the same group trying to do both things, and even from a management perspective, if you try to manage researchers alongside, especially when there's only one or two researchers alongside folks that are working on more traditional product development, it's super challenging to make that work. Because the research teams tend to really almost deliver value in spurts, there's some kind of insight that they have, and then there's a whole bunch of value delivery very quickly, and then maybe not a whole lot for a while longer. If you're trying to measure that against a more traditional software engineering, there's much more of a very defined value chain, like a developer picks up a ticket, does some work, some output comes out and delivers kind of what it said on the tin, and either works or it doesn't. It's much more binary in that sense.

Whereas I think in the data space, you end up with much more of these kind of open ended research projects where it's not necessarily clear what value delivery you'll get, and that makes it very challenging to get started. So what we actually were doing there is pulling the data out in an infrastructural or programmatic way initially, so ubiquity was basically delivered by getting as much data coverage as we could through systems that we were building, instead of going out and trying to convince teams to do anything. Only once we actually found some value in that data would we

go back to the engineering teams and then try to get them to productise it. A lot of times that would happen naturally on its own because the engineering team had noticed that they're providing multiple feeds, where now there's downstream consumers, so they wanna think about it more of an API and it becomes something that because that dependency is created and they can see the consumption, it becomes more pressing or important for them because it's now quantified.

And that's where the computational governance and lineage piece, I think comes in kind of nicely, 'cause ultimately what you wanna do basically is expose to software engineering teams, "Here's how your data is being used. Here's who's using it, here are the consumers of it, here's all these dashboards or reports that's being powered. Here are the decisions that are being made based on the dataset that you're providing." That visibility then, and especially if you can make some social pressure happen, so basically connect your consumers, the data back up to the producers, that whole cycle should create really a reinforcement structure to be able to do the organizational cultural shift toward being effective with data.

Really what we found that worked well was asking very little until there was clear value associated with it, and focusing more on infrastructure and ubiquity, and getting these computational governance systems in place and getting those feedback loops built up so that when there was an ask it was an ask for specific reason instead of an ask for enabling a research function or something like that.

Scott Hirleman
Yeah, and this is an interesting trend line that I hadn't really noticed before, but Sadie Martin on her episode was talking about measuring the value of your data projects, and that one of the things that she talked about was, especially when you think about research type projects, the value isn't in was the hypothesis correct and it had this big, massive value. The value was driving towards proving or disproving hypotheses, because when you do that, if you... Unless you're really, really terrible at coming up with hypotheses, you're gonna get better at that, and so if you get to a function where you're very, very good at doing that effectively, efficiently and all of that, you're going to end up providing those huge values, whether they're in spurts or not.

My exec sponsor has called me a "serendipity engine", and that's kind of what you need out of research, is that you're gonna end up having... It's the same thing with community management and things like this. There was one week where I created five insanely high value partnerships or whatever opportunities, and it was like, but if you try to put that on me, if you must generate these many of this type of opportunity for us, you're not gonna get anything out of that. So you do have to free up the research people to do that, but I like the way that you're talking about it. It's

almost like you have a known swamp next to your data mesh and that it's fine, and that what there is is the access by default approach to it, so you don't throw stuff with PII in there. And that you could think of that and it's like, this is the, I'll use a gross analogy. It is the kiddie pool, so you know that there's quite a bit of kids that have peed in there. You know going on in there, that it's not the highest of quality for swimming in, but that you know that that's what's going on. And so that's a really interesting potential recommendation that you have something that is known low quality, but that people can get a sense for what is that data on the threshold that the teams might think that they might wanna share, and that you create that opportunity for that.

Justin Cunningham
Yeah, yeah. I would even push into the security area of that a little bit, because I think there's a way that you can keep things secure while still enabling that broad access when you need it. And that is basically, if you can apply... Obviously, you wanna have strong permissions around anything like PII data, anything sensitive in nature, but if you can apply through a governance system gates effectively, where you can have users be able to really quickly establish that they have a business need for the data and get access to that data, that becomes incredibly valuable.

Because like I said, I think the main problem that you have in any kind of large scale data organization is access being the number one thing. Because once you can access the data, you can then determine that it's valuable and getting it cleaned up is not challenging, but if you can't access it, you can't get the initial insight. So the thing that I actually spend a lot of time thinking about is, how do you get it to be secure, but then also make it really effective for the person that needs access to that data, and make it really fast for them to be able to reach out to the upstream team and get access to it with some kind of justification? And depending on the sensitivity of the data, it might be, in some cases, okay to have the user self certify and then audit that. Instead of having them go through like a request for permission, have them say like, "I'm accessing this data for this reason," and then it goes into an audit que to be looked at in the next 12 hours. That kind of thing, I think pushing the on to the person using the data, where you've got clear guidelines and you're actually looking at it, not editing it after the fact, is a good way of actually balancing these concerns and considerations. I mean, that'll vary obviously from organization to organization in terms of how sensitive the data is, what the level of protection required is, but I think that's one of those areas where computational governance, I think is gonna make a huge difference in the future.

Scott Hirleman
Well yeah, and I'm thinking back to my conversation with Sarita Bakst, who's at JP Morgan Chase, and that's one where I think she’s not nearly as risk averse as I think a

lot of people would think of a governance person at a bank, but even she would be like, "Yeah, I don't know if I'm gonna do that." So exactly what you're talking about of, if you're in a space where it's not that big of a deal. Or you do have that like "Here's the access by default, if you need the access to the PII, yeah, you've got to self certify, and we're gonna lock down your ability to move that data out of this." You've got your sandbox, but your sandbox gets destroyed relatively quickly. Or there isn't an ability to exfiltrate data from that sandbox. So you create those things that make it so that you've got pretty good assurances and you say, "Hey, I'm trusting you. You screw up it's on your own head."

Justin Cunningham
Right, exactly. And I think there's a lot that you can do also from an automation perspective. In particular, like the sandbox thing that you're talking about, I think is a good start. Joinability restrictions, I think is another area that's pretty ripe. Where if you know some data is sensitive, restricting the joinability of that, or the writability, into other areas so that you can prevent leaks. And then understanding from a, I suppose from a data classification standpoint and from a lineage standpoint, how that data is being transformed and moved, is I think incredibly valuable. Because then you start to have a computational understanding of this data, this new dataset being derived from this existing dataset, so it should inherit the same characteristics fundamentally, it should have the same privacy or security controls.

If that needs to change, you can then go through a process of certification and make that happen. But the tighter you can get that feedback loop, the more that you can do with it, the more you can empower your developers to have freedom in that space. 'Cause ultimately what you wanna do is balance the velocity of your data teams versus all of the other considerations around governance, and making that really ergonomic, I think is kind of critical.

Scott Hirleman
Yeah, I think that Sarita Bakst talked this too, of once you really understand what data you're trying to share and why, she was talking about it for regular data products, but even in this sense that we're talking about of kind of the research angle, you're freed up in a lot more ways. If you inform teams as to, "Hey, this data could be used in this bad way. Let's be sensible around this. But this other thing where you're not sure if that's borderline, come talk to us." And then we'll wanna say, "No, it's not borderline. It's totally okay to have that." And so then it's like, "Oh, we can free up much more access and use and stuff like that," of just using that central team to stop making the decisions and start informing the ability to make those decisions.

Justin Cunningham
Yeah, yeah, absolutely. I think err on the side of caution, but build up the tooling to

make that work. One of the things that I've seen that's been super effective is just having something along the lines of the... If you've ever stumbled on a Google Doc that is locked, it'll have a big button that shows up that says "request access". Just having the ability from a data perspective, a user tries to query something that they don't have permissions to query or they're trying to access in the data catalog when they don't have permission to access it, have the big "request access" button do some checking in the background to see whether or not that user should fundamentally have that access, or it's something that requires manual review. And then automate that process, have the escalations happen to the right people without them having to get anybody involved, and you should be able to cut the time that it takes. In the same way that you can request access to a Google Doc and get that access in minutes, a lot of times. To the same kind of cycle time. That's really what I'm talking about.

Scott Hirleman
When the people who own the data are the ones making the decision that Mohammad Syed and I talked about the micro versus the macro of these microdecisions keep flowing through the data governance team and they don't have enough information to properly make that microdecision, so it goes into this backlog and it takes... Versus those microdecisions, if the teams that own that data know what the question is, boom, 85/90% of the time, they can get to that very, very quick, "Okay, yeah, we're gonna give you access." And those other times, yeah, they wanna go to the central governance team and be like, "Hey, let's make sure we're doing this in an appropriate way." But it's not that big of a risk if you're approaching it in the right way.

Justin Cunningham
Yeah, exactly.

Scott Hirleman
So we've got a little bit of time left here, but I wanted to talk about what we talked about before, which was the lineage stuff. And so you've got kind of an interesting approach. I'm just gonna let you kind of talk about that. If you want, I can tee you up with questions. But we were talking about my bugaboo around lineages, why don't we have source system data in there? We kinda talked about that immutable schemas for your, however you're moving data, whether it's streaming or it's not, like the pipelines, that there should be immutable schema, that there's a referenceability and that you don't have to have it ping that reference thing every single time, but that you are pushing that with what you're pushing. You're pushing when it's getting pushed into a new system that comes with a schema reference and somebody can look that up or however.

But I think you've got a much tighter definition and approach there, so if you wouldn't mind sharing a little bit about how you've been thinking about that and the challenges that we see?

Justin Cunningham
Yeah, absolutely. So my view on it is that we've kind of started with lineage, and I'm not totally convinced that that's actually the right starting point. I kind of think that, if you think about it, lineage and declarative configuration are effectively the same thing. One is just causal. I think as a general rule, if you start with something that's causal you'll end up getting better results. What I mean by that is, in the Netflix case, what we were trying to do basically was defined centrally, "Here's what the data movement is supposed to look like, and then we'll configure the infrastructural pieces to make that data move in that way, and we'll configure the transformations, we'll deploy the software and so on, so that you get that."

But what that gave us was basically a declarative view of what the lineage should be, and that kind of draws in maybe the confluence of the schema management system, like a schema store, like something like the confluent schema registry for Kafka, for example, with a data governance system, or data catalog with your lineage system. I think if you configure it in that way, what you start to be able to do is treat the data catalog and the data governance and lineage system more as a declarative configuration instead of an artifact. And I think if you ultimately get into the realm where you have to be able to do something like make a schema change in an upstream system and then have that flow across multiple downstream systems and understand the impact and the breakages associated, the centralisation and the making declarative of that configuration becomes super powerful. Because you can then look at it analytically effectively and say in advance, "If we make this schema change, this will be compatible with these machine systems or not, and this will be the impact on these datasets or not." And if you have an understanding of what data is being used and how, you can then kind of create communication channels really effectively between downstream consumers that would be impacting upstream producers, and back those changes in all the way up to the source systems.

So I think right now it's still early days for data governance systems, and I think that's gonna be a huge advantage going forward, if we can start thinking more about this governance problem as a configuration problem, more than just an artifact.

Scott Hirleman
Yeah, I think I hadn't thought about it that way, but we are kind of saying, "This is what it is," instead of, "This is the way it should be." Let's test to make sure that it's going to be the way that we want it to be. But yeah, Chris Riccomini in his episode talked about at WePay they implemented a system where the developers could test

in their CICD pre commit phase of, "Hey, we're testing this change," and it would be like would just kind of full on block them and say, "No, you can't commit this change 'cause you're breaking this stuff." He said 80/90% of that time it was like, "Oh, we didn't know that that would cause a problem, so let's not do that change, let's not drop that column 'cause there's no real reason to drop that column." But that 10-20% of the time, it's like, "This thing is really causing issues, or we really need to change this," or whatever, and so then that kicked off a negotiation and that kicked off... You talked about earlier, API concept. We do need to be able to version, we do need to be able to break that downstream consumption when it makes sense. Some of that is, "We've just got performance concerns and this thing is breaking our application, we definitely need to not make sure... " The data isn't more valuable than the application functioning. If the application functioning no longer happens, then the data coming from it is not gonna be very worthwhile anyway.

There's that concept of... Or if the complete business model has really changed, or the way that the data that should be shared about this application or this business, this domain has changed, and so we wanna tell those consumers, "What you've been consuming, it's not really relevant anymore. So we do need to break it simply because what you're consuming isn't a thing." But like conversation, like getting people in a room, you talk about connecting producers and consumers, people haven't done that.

Justin Cunningham
So we had a super long conversation at Netflix early on about how we wanted to deal with it, like how do you deal with these breaking changes? This is fundamentally, I think is actually an organizational cultural challenge. 'Cause where we landed on it at Netflix was, our culture is freedom and responsibility, and what that means in this context was we were gonna show you a list of what was gonna break downstream and give you a button that you could push where you could force that change if you needed to. 'Cause it's ultimately your responsibility if you're gonna make that change or not make that change. We're giving you context on what the impact of it is, but like you said, there could be circumstances. And as we were debating it, that was what was coming up, we were talking about it, it's like, "Oh, there could be circumstances where somebody really needs to make that change, and we don't wanna block it." Maybe it's better to have breakage in one area than in another, in some case.

If that's the case, we need to, as an infrastructure organization, get out of the way and let that happen, but also make it clear what the extent of that breakage is, who's gonna be impacted, what those systems are doing and so on, and understanding and owning that state essentially is almost critical in being able to actually perform that function, at least in a very deterministic way. You can certainly do it in a more

heuristic way if you have an understanding of after the fact like what's been happening, but we could say very definitively like, "If you do this, these things are definitely going to break because they're using this data."

Scott Hirleman
Yeah, I've talked about this in a lot of episodes, people talk about developers not having empathy for data consumers, and it's like they can't. We literally don't put information in front of them to be able to care. So they have to evolve their schema and they have to be able to make these changes. If they don't know what their changes are gonna do, either they can't make changes or they have to force through the changes and then downstream consumers just deal with the consequences. So we have to flip that script to be like, we have to give them the capability to care, right? If you don't know it's happening...

Justin Cunningham
There's two sides. I'd say you have to give them the capability of care, and also without understanding exactly what the lineage is, declaratively, I believe, it's hard to build the tooling to minimize the breakage that could potentially happen. If all you know is there's stuff downstream that depends on it, that's very different than knowing there's an application three layers downstream that depends on the ID of this item continuing to exist in be in it. You can change a lot about the data as long as you don't touch that ID that the downstream system is dependent on.

So really, I think the richer that you can get that metadata to be in, the more consumer level information you have, the more capable you are of managing breakage. And that breakage management, I think is absolutely critical, or will be critical in actually pushing this uptick to producers. In the same way that you've got an API and they wanna manage who's consuming these fields, and you do things like field renames and so on in order to wean folks off of deprecated fields, you've gotta do effectively the same set of things in the data space, it's just that we don't have the operational maturity or tooling around it in the same way that engineering teams are used to in the public API realm, and that's where we're missing still.

Scott Hirleman
Yeah. It's weird to me how it feels like it should be much more obvious and that we just have to deal with the semantic, but it's just not. The more that I talk to people, the more it's just like, it's just not. And I still, I can't fully understand or grok on why it's so different, but it is. It just kinda is, and I think we need to get to that tooling side more. Because these are challenges that shouldn't be nearly as difficult as they are, and tools aren't the thing that solves the challenges, they're the ones that we figure out the issues and we can leverage them to help us prevent or tackle challenges. But I'm not looking for tools to solve everything, but even if you have the right people

process, your tooling needs to be able to do things.

Justin Cunningham
Yeah. Fundamentally, where we are today is basically at the point where you'd have to have a meeting with the data team broadly and say, "We're gonna change this, what do you think it might impact?" That's not a good place to be if you wanna prevent stuff from breaking or not drive the engineers crazy if they're trying to make changes. It's much more effective if you can say, "These two fields are the ones that people are relying on, and it's for this report here, and this is the business value of it, and here's the person who owns it. You can press this button and it'll send them a notice saying that you're gonna deprecate it in six weeks or you're gonna remove it at some point in the future," and so on.

That level of automation, that's really where I think computational governance has gotta step in, and that becomes then the critical linchpin for actually making this stuff work in, work in practice. 'Cause that empathy piece, that's the organizational change component, but that doesn't exist unless you actually get people talking to each other and working together to some kind of common goal. Otherwise it's just kind of like the "us versus them" thing that kind of exists today, and that doesn't... Without addressing that fundamentally, it's gonna be really hard to implement something like a data mesh at scale.

Scott Hirleman
Yeah. Fully agree. You ask into the void, "I'm gonna make this change. Is this gonna break something?" And it's like somebody that's four downstream changes, "Oh, there's this table and then there's this augmentation and its table too, and then the augmentation." They're consuming table four or five down the stream, they have no idea that that change is gonna break something for them, they can't know that either. And so there is consumer driven testing and things like that, and it's like if they have to do a bunch of work rather than your system should tell you, "This is what this is going to break." So I hope we get there. I really, really, really hope we do.

Justin Cunningham
I think the other thing that's interesting about that is, if you think about it from a persona's perspective, the user at the very end of that is often somebody and something like the C-Suite, and the thing that's gonna break them, it probably isn't even apparent or obvious at that level. And they certainly are not gonna know about it. The breakage may not be like, "Oh, it just stopped working," it's probably gonna be more insidious, like, "Oh, We're suddenly getting results that are just slightly off, so we're just making worse decisions for the business as a whole, and we don't really understand that." But then somebody's gonna turn up six months later and downplay it and yeah, we're not in a good place then. That's how you end up with all

of these challenges too around trustability of data. If you don't manage that process well, then trying to be a more data driven organization too, becomes much, much harder. And that's in part why I say you're probably better off starting with just simpler ubiquity, and then as you develop value, like focusing in on those specific areas, getting those to be really, really solid, developing entrenched management muscle and then expanding outward from there.

Scott Hirleman
Yeah, Abe Gong talked about in his episode around Great Expectations. He talked about those expectations are the thing... So you don't have that insidious change of like, "Hey, we're expecting this to be one to 10," and all of a sudden the range is only... You might have something where you're expecting, maybe this is where we need model monitoring and things like that, where maybe that you're expecting one to ten and it starts to have things that are above ten or zero or whatever, or nulls, and it's like, "Okay, that's an issue." But if you were expecting one to ten, and you all of a sudden it's going one to five, are you gonna have that detection? So you need both of the observability side of what is our distribution of these values, and has that distribution changed significantly? Is that because the world has changed, or because we've changed the way that we're computing this and it's no longer relevant or whatever?

Justin Cunningham
Yeah, yeah. That actually reminds me of one other thing, which is, because this has come up a ton in data mesh conversations and in data conversations generally, which do you optimize for? Do you optimize for trying to get it right upfront? Or do you optimize for change? And my argument has always been that you have to optimize for change. 'Cause even if you take the time and make the investment to get all of your data right upfront, the business isn't static, and it's never gonna be static, and as the business changes, the data is gonna change and the meaning is gonna change, the semantics around it are gonna change.

So you fundamentally need an adaptive process, and if you have an adaptive process, you need to care a lot less about getting the data right initially, because you then have a process or you can iterate on it really fast. This kind of goes into what you're talking about earlier too, around research. The thing that you actually wanna optimize for, and that is learning. Are we learning fast enough? How do we get the rate of learning faster, rather than the output of that?

In the same way, I think if you get this right, you end up optimizing almost for the meta principle, like how quickly can we evolve the data into what it needs to be once we develop some insight. Rather than broadly, how much data do we have that had such and such coverage. That's much less relevant than if we find some gold, how

fast can we mine it, is a much more interesting picture, than how fast can we mine everything.

Scott Hirleman
Well, and I would even push back on change versus getting it right, and I would say change plus trustability. And so you don't need to get the data right, you just need to tell people that this is the level to which you can trust it, and so that you are optimizing for that, being able to evolve, evolve, evolve. And so that you're not spending too much time on stuff that's not of value, but that you have that like, "Okay, I actually can trust this, or I can't trust it. And I can trust that I know how much I could trust it." Mindblown kind of thing. Well, this has been so phenomenal. Sorry, we went over a little bit the agreed upon time.

Justin Cunningham
No worries.

Scott Hirleman
So is there anything we didn't cover that you think is something that people... Or do you have a button that you wanna wrap up the thing on? "Here is my exact, my philosophy." Or you've solved data mesh for folks, in a minute or so.

Justin Cunningham
I don't think there's anything like that. I would say the number one thing that I would focus on is keeping things simple and make sure that you've got value delivery. 'Cause fundamentally, if you don't have value delivery, nothing else matters. You don't need a ton of breadth for value delivery, you don't need super high quality data initially for value delivery. You need broad enough data that you can get a team of analysts in to start developing insight, and once you start developing that insight, the rest kind of follows a bit naturally. Because back to what you were saying just a second ago, if you find that data that's marked as being of low quality or you can't relate trust, but somebody finds some insight in there, then it's much easier to say, "Well, let's clean this up."

Because the clean up then is saying, "We have something that could potentially be this valuable, but we're not sure if we can trust this data." So it becomes a much more tractable bet, and I think viewing it as a series of steps that you can take to improve and make those tractful bets will be more likely to succeed, I suppose, than trying to make large scale organizational transformation without that value delivery super clearly defined.

Scott Hirleman
Yeah, and I think this is the common through line for almost all the conversations of

the people who are having success with data mesh implementations, is that "Think big, start small, move fast" type of concept, but really what it means is like you can build muscle, you don't have to get it perfect. You don't have to nail it. You're finding places where you can have a high return on investment because you've got a low investment, and so you might have a moderate return, but you don't go for the thing that has the massive, massive potential return, but the investment is twelve months and all that. You start to figure out how to do this and that you can make this repeatable, and that then you can start to chunk it up, but you start to tackle more and more difficult challenges.

But until you figure out how to do this, you're just gonna fail if you just try and run headlong into your most difficult challenge. So yeah, I think this is really helpful in a lot of that. And I think especially somebody who's done it in organizations who haven't been focused on specifically the "definition of data mesh", of how do we actually get to an organization where we can achieve what data mesh is trying to achieve?Data mesh, I don't care if you call it data mesh or whatever. I tell people literally, if they're talking to other orgs to call it unicorn farts. So they get "data mesh" out of their mouth, that they're not trying to go and say, "We're gonna do data mesh and data mesh solves our problems." It's, we're gonna put the resources and the capabilities in the hands of the people who know the data best to be able to deliver it in such a way that people can understand it and trust it and use it, and then we're gonna give the people who want to consume that data the capability to understand and trust that data and then access it and use it and actually do that. And we're gonna raise the bar of those people, and we're gonna make it so that we're not locking ourselves into the enterprise data warehouse model and stuff like that.It's funny how much, even Zhamak and I both talk about how data mesh is trying to achieve something. If you're just doing data mesh for the sake of data mesh, you're gonna fail.

It's an approach. And we're still figuring out exactly how to do it, right? We're figuring out those good ways, in those bad ways, and that's kind of the point of the podcast, is to extract this information of what have people learned? What have they seen work, what have they seen not work? And move from there.

Justin Cunningham
Awesome. Makes total sense to me.

Scott Hirleman
So if people wanna follow up with you, just in general, the best place to find you, Twitter, LinkedIn. Where do you want people reaching out?

Justin Cunningham

Yeah, LinkedIn would definitely be the best place.

Scott Hirleman
Okay. What would you want people following up with you about? That's a broad question, but what do you find the most interesting? Or is there anything in here where you're like, "I really want people to follow up with me about X, Y, Z topic."?

Justin Cunningham
If folks have interest in the data governance topic, I'd love to talk more about that in particular. In general, I suppose just challenges in the data space. I'm always interested in challenges that companies are running into in the data space, especially with implementing this stuff. 'Cause I've done the implementations a few times, and some of it I know is super hard to build out. So where folks are getting stuck, I think is an opportunity potentially to rethink how you do infrastructure in that area, and that's an area that I'm personally interested in at this point.

Scott Hirleman
Awesome. Okay, well, this has been so great. I think it's gonna be very useful in framing how people can think about these things and what you've seen has worked. So I really wanna thank you for the time, and thanks everybody for listening. I'd again like to thank my guest today, Justin Cunningham. As per usual, you can find a link to his LinkedIn page in the show notes. Thank you.

Thank you so much for listening to this episode of Data Mesh Radio. Hopefully, it was useful to you. If you'd like to connect with the show, you can find us on LinkedIn or Twitter. If you'd like to connect with me, you can do the same. If you have feedback or especially if you'd like to be a guest, we've got some links in the show notes to tell you how to do that. I'd love to hear what questions people have and how I can be useful. And then this is provided as a free community resource by DataStax, so please do stick around after the music if you'd like to hear more about what DataStax is offering with their serverless managed version of Cassandra.

Data Mesh Radio is a free community resource provided by DataStax. If you're not aware, I'm actually employed by DataStax. And they've allowed me to be a resource for the general data community, learning about data mesh, data as a product, and other related topics. If you're not aware, DataStax has made a lot of pretty big interesting changes in the last few years, and now offer a multi region, very performant serverless database offering with a number of easy to develop for APIs with like GRPC, JSON, GraphQL, REST. If you'd like to learn more, check them out, and you can check out their AstraDB offering, and you can use the code DAAP500, that's DAAP 500, or data as a product 500, to get $500 of free credits. You can also check out what they're doing with streaming. It's pretty interesting.