EPISODE 814

[INTRODUCTION]

[00:00:00] JM: Lyft is a ridesharing company with petabytes of data. Within Lyft, many different employees can use those datasets to build useful applications. A business analyst creates a dashboard to see how driver satisfaction is changing over time. An economist studies the pricing data to ensure that Lyft's prices are competitive. A data scientist creates a report of how the speed of a ride correlates with five star ratings. A machine learning engineer trains a model to detect fraud on the platform.

All of these use cases makes sense, and in each of them, the employee at Lyft needs to find the necessary datasets within the company to build their application. Amundsen is a tool for finding and discovering datasets within the company. Tao Feng and Mark Grover are engineers at Lyft and they join the show to talk about the problem of data discovery and the tools they have built and open sourced at Lyft.

A few updates from Software Engineering Daily land. Podsheets is our open source set of tools for managing podcasts and podcast businesses. A new version of Software Daily, our app and ad-free subscription service, has been created, and Software Daily is looking for help with Android engineering, QA, machine learning and more. We definitely need help finding all the bugs in the mobile apps. Not to say the mobile apps are totally broken. They just have some bugs, and we'd really like to iron them out so we can create a better experience for the listenership.

The FindCollabs hackathon has ended, at least the first one. Winners will probably be announced by the time this episode airs, and we'll be announcing our next hackathon in a few weeks. So please stay tuned. The links for all these updates are in the show notes, and let's get on with the episode.

[SPONSOR MESSAGE]

[00:02:03] JM: We are all looking for a dream job, and thanks to the internet it's gotten easier to get matched up with an ideal job. Vettery is an online hiring marketplace that connects highly qualified job seekers with inspiring companies. Once you have been vetted and accepted to Vettery, companies reach out directly to you because they know you are a high quality candidate.

The Vettery matching algorithm shows off your profile to hiring managers looking for someone with your skills, your experience and your preferences, and because you've been vetted and you're a highly qualified candidate, you should be able to find something that suits your preferences.

To check out Vettery and apply, go to vettery.com/sedaily for more information. Vettery is completely free for jobseekers. There're 4,000 growing companies, from startups to large corporations, that have partnered with Vettery and will have a direct connection to access your profile.

There are full-time jobs, contract roles, remote job listings with a variety of technical roles in all industries and you can sign up on vettery.com/sedaily and get a $500 bonus if you accept a job through Vettery. Get started on your new career path today. Get connected to a network of 4,000 companies and get vetted and accepted to Vettery by going to vettery.com/sedaily.

Thank you to Vettery for being a new sponsor of Software Engineering Daily.

[INTERVIEW]

[00:03:47] JM: Tao Feng and Mark Grover, you guys are engineers at Lyft. Welcome to Software Engineering Daily.

[00:03:51] MG: Thank you for having us. We are super glad to be here.

[00:03:53] TF: Thank you.

[00:03:54] JM: I want to talk to you today about data discovery, and in order to get to the topic of what that term actually means, let's just talk about how data science works at Lyft. So as I understand, most of the data that data scientists are interfacing with at Lyft is not data that's created by the data scientist.

It's created by production services, like a marketplace service that's setting the price on rides that people are taking, or perhaps telemetry data for how people are traveling through space, and this data needs to be further processed in data science jobs.

Describe how data gets created at Lyft.

[00:04:37] MG: The source of data at Lyft is one of three different things. The first one as you were alluding to, Jeff, is from the apps or services that send events. So say you want to take a Lyft ride, you open up your app, and then you go through the "funnel". You select your destination and you select your mode. Each of those clicks or actions is going to send an event which we track and then later can be used for analysis to see how well that flow works. So that's type one.

Type two is what we call CDC, change data capture. So we are taking replicas from our online production systems and then bringing them into our warehouse in real time. Type three is any external data or third party data we may get from vendors that we upload on the side to our data warehouse, so folks can do analysis later on. So those are the three different sources of data.

[00:05:29] JM: This data can be used for many purposes. It can be used to build machine learning models, to build dashboards. Describe some of the use cases for building things with these large datasets that are getting created at Lyft.

[00:05:43] MG: Yeah, absolutely. I think the first and foremost use cases, the first one they set was dashboards, and they are used to make decision, decisions as big as areas where the company needs to invest more in, to more day-to-day decisions around how much money you could spend on acquisition or engagement, stuff like that.

The second is also what you alluded to, which is around modeling. The price that you see in the app or the ETA that you see about when your driver will be available are based on machine learning models, and it's the same data that powers those models.

The two use cases that are in addition to these two that are unique to Lyft, the first one involves an area of the organization we call the operations. So these are folks in various different hubs that are responsible for the health of the marketplace. So they want to make sure that there are enough people driving at the right places at the right time for the rides that are going to be requested at that time.

These people need tools in real-time to control the health of the marketplace, maybe incentive drivers to go to a certain place at a certain time and so on. That's the third use case. The fourth use case are folks who are experimenting with the Lyft apps or the services in deploying new code.

So it could be your traditional idea of experimentation around here's a red button, here's a blue button. Which one works better? But at Lyft, we are also doing all kinds of experimentation around here's a new version of the pricing algorithm and how would the marketplace dynamics behave for such an algorithm? This list is not exhaustive, but those are the top four that come to the mind.

[00:07:16] JM: There's a variety of data roles at a tech company like Lyft. I've had conversations with people from Netflix and Uber and other companies where they talk about the fact that you've got this explosion of roles that are related to data. You've got not only data scientist and data engineer, but you've got the data analysts. You've got the machine learning engineer.

You might have business development people who need some data and perhaps and they know some SQL. Describe the different data roles at Lyft. Who needs to be able to find and access large datasets?

[00:07:54] MG: Yeah. So their are official titles, may range anywhere from analyst, to local analyst, to a local analytics manager, to a data scientist, to a research scientist, to a product manager or an exec.

So I think perhaps one way to look at this problem is to list out all these titles, and most of them have a responsibility around using data that's curated and trustworthy to make decisions. That's perhaps the most common use case that's consistent across all those folks.

Then the second large category of roles around research scientist, is what we call them here at Lyft, as well as machine learning engineers, is around building and productionalizing prototypes or building, prototyping, productionalizing models that then power some very salient features of the app, like pricing and ETA.

[00:08:44] JM: Explain what data discovery is.

[00:08:47] MG: The idea of data discover is that for folks – let's say you come in and you're a new analyst in the organization and you want to figure out how are we doing with cancellations, right? If you are fairly new to the company, you don't have the tribal knowledge to even know where to start looking.

Like what is the table that you need to run a query on, or maybe you don't even need to run query, because somebody has already done this analysis. So data discovery is this problem of discovering the right trustworthy sources of data, so you can gain reliable insights from it.

[00:09:25] JM: Data discovery is important to all of these different people within an organization who might need to access data. If I'm building a dashboard, I need to find the dataset that's going to power that dashboard. If I'm building a machine learning model, I need to find the datasets – as many datasets that would be useful to building a machine learning model. Oftentimes, if I'm building some data-driven solution within a big organization like Lyft, I don't sit anywhere near the person who is actually generating that dataset.

You may have somebody in the Seattle office who writes the service that calculates prices across the world. Then in the San Francisco office, perhaps you have somebody who wants to

build a dashboard for how prices have moved over time. So you have this – the person may not be even be in the same sector of the company. So you have this problem where people in different parts of the organization need to just be able to discover and access datasets that they don't even know exist or they don't know where they are.

How has data discovery historically been solved at Lyft before you guys started working on the solution that we'll get to?

[00:10:44] TF: Sure. So in histories like analysis or a data engineer may ask questions in Slack or places saying, "Could I use this table to solve, for example, getting the ETA information or getting the pricing information?" Some of the domain experts which have the knowledge will answer these questions. Then we have seen a pattern. So this kind of question has been sometimes repeated asked.

Also, there are some fragmented static Wiki page may document some of these informations. Traditionally, these area data history is bit fragmented at Lyft before Amundsen.

[00:11:26] JM: How does that fragmentation create problems within the engineering process?

[00:11:32] MG: Yeah, I think the biggest problem it leads to is a lack of productivity. That means that there's a lot of tribal knowledge in people's heads, and the amount of tribal knowledge you have is obviously proportional to the time you spent at a company.

For a company that's growing as fast as Lyft, that just doesn't work. So the biggest problem is around productivity and also around democratization. So if we were to dive deep into what that word means, it's really like democratization is that there shouldn't be any classes between people. It's a classless society.

Without data discovery being solved, we noticed that there were a lot of classes being created. The very first and obvious one was around the people who have been around for a long time and the people who just joined, and these people may be equally smart, but the people who have been for a lesser amount of time aren't as resourced or enabled to be productive, and that was really the problem that this was causing.

[00:12:32] JM: When did you start trying to build a technical solution around solving this data discovery problem within Lyft?

[00:12:40] MG: Yeah. We were doing some interviews about a year or a year and a half ago when hearing from our users that there wasn't a standard way to discover the data around them. That led to us doing some prototypes. They were started around June of last year, I'd say.

[00:12:57] JM: When you started attacking this problem, what were your initial thoughts on how you might solve it?

[00:13:01] MG: For that we actually ended up talking to a lot of our peer companies. We felt that this was a problem that existed in other places too and we were lucky to have contacts at other companies.

So a few companies that we talk to were Facebook, who had this solved this problem using a tool called iData that's internal to Facebook. Airbnb had a tool called Dataportal, which they built. LinkedIn, where Tao used to work actually before he came to Lyft, had a tool called Warehouse where they try to solve.

So one of the first things we did when we first got a hang of this problem was we wanted to go and talk to all these organizations, and that's kind of how we got started.

[00:13:38] JM: I did a show a while ago with the individual who started cursor.com, and I think that was based on LinkedIn's data discovery platform. Is that right?

[00:13:47] MG: That's possible. We aren't familiar with Cursor.

[00:13:50] TF: Yeah. I'm not either.

[00:13:52] JM: Okay. All right. Fair enough. Well, focusing back on Lyft and your data discovery solution, how is data stored within Lyft? So when we talk about, for example, a marketplace

service that is storing a recollection of all the different prices that have changed over time versus talking about something like, how much did rides cost over a given period of time.

Are those data storage systems going to be consistent from service to service?

[00:14:25] TF: At Lyft, there are two type of data store. One is for into the online data store. One for on the offline data store. For those, for example, marketplace or location or ETA, they will send a real time data into our online data store, mostly Dynamo or Mongo, MongoDB.

Then once those raw data is stored, some data engineer or some pipeline will kind of populate or replicate these type of raw data into our offline data store, mainly Hive and RedShift. Then data engineer will build the ride table on top of this raw data to get the ride table.

We also have another one like in-memory OLAP, called Druid. So it's more for faster query analysis.

[00:15:11] JM: Are the Druid data stores and those databases, those tables that the data engineers are building off of that offline data store - is the entire offline data store in HDFS, or what's the storage medium for the offline data store?

[00:15:29] TF: Mostly S3.

[00:15:30] JM: S3. Okay. Does anybody get access to that offline data store? Is it totally accessible or is there some kind of security policy around who can access what dataset within the offline data store?

[00:15:42] MG: We obviously have an obligation to safeguard the data that users send us. The first and foremost thing we do with stuff that's super sensitive doesn't even come to the online data store.

So it obviously is in online data stores, but it doesn't come to the offline data stores. Then from the data that's in the offline data stores, there are various different policies that integrate with an in-house service that we call the ACL service.

So this ACL service sits in between the BI tools or the dashboarding tools that are used to surface most other data to the larger company as well as the databases that implement access control. So they're able to sort of lockdown both the access control on the database level as well as the BI/dashboarding level.

[00:16:27] JM: ACL? Does that mean access control logic?

[00:16:30] MG: Yeah, access control list. Yeah, exactly.

[00:16:32] JM: Access control list. Okay, cool. So you have mechanisms of controlling who can access what dataset.

When we talk about this offline data store, we've got a bunch of data there, and we can materialize different views on that big data lake so that we can perform analytics and queries on that data faster than if we were just accessing them in raw S3 buckets.

Of course, again, we have to figure out what is actually in this huge data lake that makes up our offline data store. So if I'm a data scientist and I want to be able to easily find datasets that are relevant to whatever I'm studying or whatever machine learning model I'm building, what's the ideal experience for my process of data discovery?

[00:17:21] MG: Yeah. That's a great question. We think that the ideal experience – actually, it's a few different kinds of experiences based on what you're looking at. So the first kind of ideal experience, one, is that you have a certain keyword in your mind, right? Say, I'm looking for cancellation rate or how the ETAs have been performing. So you want to be able to search using this keyword for relevant pieces. So that's the first use case.

The second use case is you just joined the company and you want to browse around what's available for data within your team. Maybe you want to look at like the tech lead of the team who you just joined so you can see like what they are using and learn from it.

So within these cases, let's dive deep into the first one, the search one, because I think that is the most common one. So the ideal use case for the search one would be you come in and you have a search engine like interface, the way you search web in the data space. So you come in, you type something, let's say cancels. Once you type it, you get a list of results.

Now, these results are of various different kinds. The first kind is tables, and you see them ranked in a similar order to how page rank, for example, ranks your website results. So you see them rank based on their usage, and usage here means query volumes. So tables that are queried higher show up higher in the query results.

In addition to just seeing tables, you also see dashboards and notebooks and work that people have already done, because the best kind of work that you can do is work that already someone else has done. So you would see that in a surface.

From these results, you are able to then click on it, dive deep on it. Let's say you aren't able to find any relevant analysis. So you have to write your own SQL query. You click on the table and then you're able to see in great detail what this table contains. So what that means is you have information about a description of the table along with all the columns and the types. But not only that, you have a description of each of the columns and you have advanced stats on the column.

So like how many records does this column have? How many column distincts? If it's an int, what's the min and the max? If it's a string, what is the average length? This kind of stuff can really help analysts, data scientists, research scientists, really figure out what we call the shape of the data and it helps them gain that trust.

In addition to that, often you want to see like who owns this. When was this table last updated? When is the first data for which this table has data for? What is the lineage, which is where is this table coming from? Where is it going to? Who else is using this table?

Lastly, you may want to see a small preview of the data to actually make sure this kind of makes sense. If this all makes sense, you may also like in an ideal world have a marker. If you go to

kayak.com, there's a 95% change that the price will go up in the next 7 days. Why can't all our data resources have a marker which says there's a 95% trust in this data.

So you have this page which really hones in on can you trust this data for the analysis you're about to embark upon. If the answer you conclude is great, you click on this button at the very bottom that says, "Go explore," and that takes you to an exploration tool where you just write your SQL queries or not. But that's really the ideal data discovery experience that we're thinking about.

[SPONSOR MESSAGE]

[00:20:52] JM: On Software Engineering Daily, we've had several shows about the future of technology education. We believe that boot camps are an efficient, cost-effective way to become trained for the tech industry. Whether you want to be a programmer, a data scientist, or a designer.

Flatiron School can teach you the skills you need to build a career that you will love. Flatiron School has immersive programming courses on JavaScript and Ruby, everything you need to become a full stack developer. If you're interested in becoming a data scientist, Flatiron School has courses on Python, SQL and machine learning.

You can learn in-person or online and you can find everything you need to get started by going to flatironschool.com/sedaily. Flatiron School has options to save money on the program, such as gender diversity scholarships and income share agreements. Flatiron School also helps the students who graduate find a job. Every graduate is paired with a dedicated career coach so that they can find a job or their money back.

The complete details are at flatironschool.com/terms. Flatiron School is a cost-effective way to start working in a tech industry. Learn more at flatironschool.com/sedaily.

[INTERVIEW CONTINUED]

[00:22:22] JM: As you mentioned a little bit earlier, there are some tools for this data discovery process already. There are things you can take off the shelf to help you index and extract the metadata from the datasets that exist across your organization.

So these other tools for data discovery - because this problem is not necessarily new, in what ways were they insufficient? Why did you have to build something from scratch?

[00:22:51] MG: Yeah. We did a thorough deep build, buy and adapt analysis when we embarked upon this. Adapt, I want to talk a little more about it. Adapt is this new criteria, which is you can take an existing open source project and adapt it and make it better and use it. So the few tools we looked at back then were vendors like Alation, as well as open source tools like Apache Atlas, as well as learnings from existing tools that other companies build. So Airbnb's Dataportal and LinkedIn's Warehouse, were the biggest ones.

It also helps for us to share some of the criteria we used for evaluation. So the criteria that we used – by the way, all of these is also in a recent presentation that we did. So you can find that on SlideShare, and we can share the link later on too. But the criteria at a very high-level was being able to support what we called search queries, which is a very self-exploratory. The second one was lineage queries, and the third was network queries, which are around who's using this and what are the common users and so on.

In addition to that, we needed some tactical support around RedShift tables as well as high tables, and we had a preference to it being open sourced. Based on that – and there's a whole metric - matrix. There wasn't really any tool that worked for us. What we really – If there would have been an open source tool that really did a fantastic job of this problem of data discovery, that would have been the ideal choice. But a year, a year and a half ago, there was nothing that really fit the bill.

[00:24:21] JM: Before we get into architecting the solution of Amundsen, which is what you built for data discovery, let's make sure that we don't lose the lead here. So you wanted your data discovery tool, the one that you were going to build, you wanted it to solve a set of problems for the different people across your organization. What were the problems you wanted to solve with a data discovery platform?

[00:24:49] TF: The problem we tried to solve is mostly our mission for Amundsen mostly like democratize the data discovery and remove the tribal knowledge. We want people to make data discovery easy and use it successfully for the metadata, for data.

So Amundsen evolved, start as a data discovery tool. So later on, we evolved as a metadata engine. Meaning that we are right now using internally for CCPA, GDPR purposes. We also use as a meta engine for other machine learning platform to ingest their metadata into Amundsen. So it became a metadata lake for the code or the data being Lyft.

[00:25:31] JM: When you say metadata here, what are you referring to?

[00:25:34] MG: Yes. When we say metadata, we actually refer to three different kinds of metadata, and it's a terminology we borrowed from a paper in this space called the Ground Project, or the Ground Paper.

We refer to three different kinds of metadata. The first one – and we call this the ABC of metadata. A is application context. This is information that you and I and everyone needs in order to just do our job. This is what data exists, where is it and what does it mean?

The second one is B, for behavior, which is who's using it? What applications are built on it and so on? The third is C, for change. How is this dataset evolving and why was it evolved? Why did someone change the type of this column from string to end?

[00:26:17] JM: So this is metadata about the databases and the datasets that are across Lyft? Is that right?

[00:26:26] MG: Yeah. Those ABCs are types of metadata. Then on the other side of this axis are the various sources or resources for which we have this information. So we can gather the ABCs of metadata for tables, but not just that. We can gather them for dashboards. You should be able to discover dashboards or reports that were made.

Not only that, you should be able to discover streams or Kafka topics that were created, or events schemas that are out there in your organization as you were instrumenting a new event, you can look at existing events that exist.

The last one, which we think is super critical as well as people, human beings, we all are in some ways data assets. The fact that you use a certain table or bookmark a certain table, or created a certain table is information that's relevant to people.

So not only all these sort of abstract resources, we should index people as well in this graph. When you search for something like that information represented in a data graph is the most relevant to expose to our users.

[00:27:30] JM: We talk about all these different datasets within an organization. We've got datasets of all the different drivers in – let's say like a dataset of drivers, and the dataset of drivers has fields like maybe – I don't know, their age, where they are driving around in. How many hours they drive on average.

Those details, those would be things that you would want to index as metadata, because when somebody is doing data discovery and they're looking for the right dataset to base their driver satisfaction dashboard on, they need to be able to search all the datasets. They don't necessarily need to search all the data quite yet. They just need to find that dataset.

So that's what you're trying to do here by extracting the metadata from these different datasets.

[00:28:29] MG: Yeah, and we think that the holy grail of next generation of data applications is actually on metadata. Whether that's on – like Tao was referring to earlier, like we started off with a data discovery application, and that's obviously the core of the conversation today. What we have realized is that the same metadata we can use to comply with regulations with GDPR and its younger, sort of California brother, CCPA. We are using Amundsen's backend metadata for compliance purposes.

Later on, like we see new applications emerge around downstream analysis. If you want to change like a certain field or a certain column who all will be impacted, and then that's all powered by the same metadata as well.

[00:29:12] JM: You also mentioned something pretty interesting, which is the metadata of who accesses different datasets. So if Stacey goes and builds a dashboard based on driver satisfaction, she's probably touched the driver's dataset, and you may be in a totally different part of the organization and you may want a driver satisfaction, a dashboard as well.

If you go and look at the driver satisfaction dataset in an ideal world, there would be some trace that Stacey has accessed that dataset. Actually, I guess, in the ideal world, her dashboard would be index. So you would be searching for driver satisfaction dataset and you might come up with, "Oh! Oh! Stacey already built this dashboard that I'm actually trying to build. So I no longer need to build this thing."

Here, you're going towards like reducing duplicated work.

[00:30:10] TF: That's correct.

[00:30:11] JM: And I also like the idea of reducing duplicated work by like indexing the Kafka streams or the other kinds of like computation that's munging this different data, because from what I can tell, most organizations have so much duplicated work.

Do you have a sense for how much duplicated work is going on in the world of data science at Lyft?

[00:30:33] MG: We were laughing when you asked that question. We don't have a sense. The closest we get to that is just the lack of productivity, and that was a qualitative survey we had done around how much time were data scientists and analysts spending on various parts of their workflow. So their standard workflow involves, first, understanding the problem.

Mostly in English in conversations with other folks but then it starts with discovery, then you go and analyze and explore, and then you dashboard and share. We noticed that a lot of scientists

during this workflow, they were spending about 33% or 35%, somewhere in there, about a third of their time data discovery, before this tool. So that gave us a lot of signal to try to solve that problem with a lot of priority.

[00:31:19] TF: So this bring us an interesting point on Amundsen. So Amundsen tried to service the most important metadata for user to use. For example, where you mentioned, Stacey is a driver analyst expert. Your dashboard that a lot of people has to use. How do we make sure that this dashboard has been serviced as a first or second entry when people use search?

So at Lyft, we have a lot of query law, audit laws about which dashboard, which dataset has been used by how many people. We get this information and indexing Amundsen and make sure the search ranking is actually fall into match this expectation. Basically, when users try to search a table, a dashboard, so the one that surface on the top entry will be the one that have been used by most people.

[00:32:10] JM: As we're talking about extracting metadata from datasets, if you've got perhaps S3 buckets in the offline data store, or you've got materialized views on top of that offline dataset, you need to be able to understand the database columns. You need to be able to understand what data is in a given dataset. That's to the metadata point.

So this brings up this interesting data engineering problem of Amundsen. Because in order to allow for data discovery, you need to crawl the metadata of each data storage system that's in this offline data store, or the materialized views of the offline data store and you need to periodically be indexing the stuff that you're grabbing.

This seems like kind of a challenge, because depending on what people like are storing their data in, it's like Cassandra, ElasticSearch, PostgreS, like Aurora. I don't know to what degree you have sprawl of different data storage types, but I imagine this is kind of an annoying challenge.

[00:33:24] TF: That's correct. When we're building this system, we have put a lot of thought on how do we get this metadata. There are two ways – two approach. One is what we call a pull-based approach. One is called a push-based approach.

Pull-based approach is like what you said, like building a column. Basically, meaning whenever there's a change in the source, we don't get it in real-time, but we build a caller to fetch this information in a fixed schedule and persisting into our graph.

While push-based approach is like whenever the source has a change, you also send a notification, an event to a stream of Kafka topics and downstream those real-time systems, a streaming system, to persist this data into our graph in real time.

So when we start building a thing about this problem, at that time, Lyft doesn't have a very mature Kafka or messaging system set up, and we don't – and it's hard for us to design a generic data model as for push model, because like you said, there are so many different kind of data source with different form. So we start with the pool model, like building a caller for some of the data source.

So we tried to make it extensible. So we build a library, a data ingestion library called Data Builder, which is inspired by a project, a popular project called Apache Coupling, which is data integration framework. So it has four phases, like extractor, transformers, loader and publishers. So we generic this caller design into four phases to make it easy to extendable for different new data source.

[00:35:18] JM: The process of crawling all these different data sources, you used Apache Airflow for the workflow scheduling. Can you describe why Apache Airflow is useful? I guess let's just talk about this engineering problem of extracting metadata from different data sources in order to build an index of the data sources across Lyft.

[00:35:41] TF: Apache Airflow is a very popular workflow management project. It's known for dependency management. It handle like schedules, certain job in a fixed schedule library. So if you think of calling the data source as – from this point of view, is that we want this kind of job to kick off in certain sequence.

We want this to say we get the metadata for all the table detail first. Once that is done, then kickoff the extended metadata. It could be, for example, the statistic for this table. Who used this

table? What is the high watermark or low watermark of this table? Then there's certain sequence. Once we build each of these as an Airflow task in our deck, so that Airflow will kick off in a certain sequence.

[SPONSOR MESSAGE]

[00:36:44] JM: When I talk to web developers about building and deploying websites, I keep hearing excitement about Netlify. Netlify is a modern way to build and manage fast modern websites that run without the need for addressable web servers. Netlify is serverless. Netlify lets you deploy sites directly from Git to a worldwide application delivery network for the fastest possible performance.

Netlify's built-in continuous deployment automatically builds and deploys your site, or your application, whenever you push to your Git repository. You can even attach deploy previews to your poll requests and turn each branch into its own staging site.

Use modern frontend tools and site generators, like React, and Gatsby, or Vue, and Nuxt. For the backend, Netlify can automatically deploy AWS Lambda functions right alongside the rest of your code. Simply set up a folder and drop in your functions. Everything else is automatic, and there's so much more. There's automatic forms, identity management and tools to manage and transform large images and media.

Go to netlify.com/sedaily to learn more about Netlify and support Software Engineering Daily. It's a great way to deploy your newest application, or an old application. So go to netlify.com/sedaily and see what the Netlify team is building.

Also, you can check out our episode that we did with the Netlify CEO and founder, Matt Billman. That was a really enjoyable episode. I'm happy to have Netlify as a supporter of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:38:39] JM: The architecture of Amundsen is worth exploring in more detail now. Now that we've laid out the problem of data discovery, we've talked about the fact that, really, the core of the data that you need to be indexing in this system is the metadata or other data sources. If people are confused about that, maybe just like rewind 20 minutes and kind of like listen again, because I think it'd probably be confusing to some people.

But if you're following what's going on here, you realize that we need to figure out what kind of backing store are we going to use for all of this metadata that we're gathering from our different datasets. So how did you look at that problem? When you've got these different datasets, different tables and you want to be able to index them, how did you chose your data storage system?

[00:39:35] TF: Sure. So to answer this question, let's do a recap of what a typical Lyft data workflow looks like. So like Mark has mentioned before, the typical flow will be like – application could be Lyft mobile clients in certain location or ETA information to a Kinesis stream or Kafka topics.

Then we have a streaming in just system. We'll get this information and persist into our raw high table. Then some analysis of data engineer will build some dataset on top of this raw table into a derived table. Then later on we found a certain use, so we used this table on a certain column. If you think of that this actually is a graph to model the whole flow, is each of the component will be a [inaudible 00:40:27] and connect them will be edges. So this is very natural to model as a graph.

There's a couple of other offering, database offering in the industry, like the first category would be a NoSQL database like the DynamoDB or Cassandra. Those are very good for performance, but they don't have a good support for join, which is critical to model this relationship. The other will be like the traditional RDBMS database, Oracle or PostgreS or MySQL. They do have choice support, but their performance is not very good. Once you have the number of nos has been increasing to certain scale. In the end, we chose graph database, and we [inaudible 00:41:09] as one of the most popular offering in this domain as our backend data source.

[00:41:15] JM: Okay. Wait. So why a graph database? Talk about that in a little bit more detail. Why is a graph database the right choice?

[00:41:22] TF: So we think like graph database could help us to easily model our data flow. Our data flow actually is like graph. Application connected with the events, because this event is fire or sent by this application. For example, column will be an extend metadata of this table. You could connect is edges say as column. Then the column is used by this user. So it also have a direction edges.

[00:41:55] JM: Sorry if I misunderstood this, but how are you finding connections between these different data sources? Let's say one person in one of part of the company makes the drivers – like the driver satisfaction database. The driver satisfaction database is like we've been sending these surveys to drivers, and we've been recording their results. That's how we accumulate the driver satisfaction database.

Then another part of the organization, you have like driver statistics, and it's like the drivers have been earning money over the course of a period of time. It seems like these two tables – well, I guess they're closely related enough. If you just have like driver as a column in each of those, you would want an edge between those two dataset nodes in your graph database.

[00:42:51] TF: That's correct. So when we start building Amundsen, our graph always keep evolving.

[00:43:00] JM: It doesn't have to be perfect.

[00:43:02] TF: Yeah, it doesn't have to be perfect. We start as a subset of the graph and we keep that thing like domain knowledge on how with different user, insight on how the graph should evolve. Then we figure out actually there is certain metadata or we should index into a graph to make our graph more complete, more searchable, more usable for people for data discovery.

[00:43:23] JM: What a great insight.

[00:43:24] MG: Yeah. To pile on to what Tao just said, I think the few sources which we scraped is the extra node, the expansion of the graph that Tao was talking about, one expansion we did in the second phase of the project was a link between a table and the Airflow DAG that created that table.

Another expansion we did was around lineage, which is like we look at a query and are able to decide that table B was constructed from table A, and hence there should be a link between those two. Now we also have a link between a human and a table. So if you ran a query there as a link of used by between those two entities as well.

[00:44:04] JM: The other component of Amundsen that we haven't really talked about is Elasticsearch. So basically, if I'm a user, I'm looking for the right data sources to build a dashboard or to do some data science stuff, I'm going to enter in a search query. I'm searching for a driver's dataset, and that's going to hit Elasticsearch.

Tell me about what you had to do to get your Elasticsearch index built properly.

[00:44:33] TF: So we mostly rely on certain audit log. So, for example, in our case we're using the Presto audit log to figure out which table or which column has been used or total use or –

[00:44:46] JM: Is it the Presto audit log?

[00:44:48] TF: Yes. So to get a sense about which table in this place has been used by more people. Then we build a certain format to index this information into our Elasticsearch. So, for example, a certain row or a certain column has certain ranking based on the usage, then when we surface this information, we use this rank.

[00:45:14] JM: That's so creative. These other data discovery platforms that are already in existence, did they take into account this kind of stuff? Basically, the popularity in terms of who's accessing the data a lot.

[00:45:30] MG: I think some of them did, though that was not like a super important criteria for us. We learned the power of this from, actually a tool that already existed, Airbnb's Dataportal did this, and we thought it was like a phenomenal idea and we just copied it.

[00:45:47] JM: I assume there must be like kind of a power law distribution among what are the best datasets, and it would almost be cool to like – you log in to the platform and just like, "What are the most popular datasets within the organization?" That's probably a great use case.

[00:46:04] MG: Yeah. In fact, the first page of Amundsen has a search box as we described earlier, but actually also has popular tables page underneath it. So if you don't know what you want to search for and just it's your first time just browsing data, you start with those top five tables. That's a dynamically populated list based on what's popular in the company.

[00:46:26] JM: So if I understand the architecture correctly, if I start using Amundsen and I do a search for like – I'm looking for datasets that involve drivers for Lyft. I search drivers. That Elasticsearch query, it hits Elasticsearch, with the Elasticsearch index, and that index has been built based off of the data not necessarily in the graph database, or it's like it combines the graph database and the Presto audit log -

[00:46:56] TF: So the search index is built in our offline column, basically our data builder. So we build [inaudible 00:47:04], like we generate a lot of the metadata into our Neo4j. Then one of the last tasks will be fetching this information from metadata graph. At that time we already have the usage information, and then build the index in an offline fashion into our Elasticsearch.

[00:47:23] JM: Okay, I think see. So basically, you have this data builder – well, not ETL job. Data engineering job, the series of Airflow jobs. The data builder is scanning all the datasets in the offline data store. It's extracting the metadata from them. It's collecting the metadata into the graph database. You build the graph database and then you take the Presto audit log separately.

Again, Presto is something we've done a show on. We've done a couple of shows on. It's like a way to query large datasets. I don't know if that's poor condensed version of it. But you can use the audit log to sort of see who is accessing what dataset and how popular certain datasets are.

So you use the data from the audit log together with the data in the graph database that you just built to build the search index that users actually will be using to solve their data discover problem.

[00:48:20] TF: Yeah. So that's correct. So currently the index is built offline, but we are moving towards this also have a real time. So meaning, like when user is doing some searching, it will also affect our Elasticsearch index. But that is still in our roadmap.

[00:48:38] JM: Cool. Just to drive this point home. Take me through the life of a query. When I am on the backend – or what is happening on the backend when I'm grabbing a dataset from the frontend?

[00:48:49] TF: When user type a query term could be a driver. So first of all, let me recap a bit on Amundsen's architecture if we didn't discuss. So Amundsen have a stream microservice. It has a frontend services. It has search service. It has metadata service. The metadata service is powered the metadata request from the frontend. The search services powered the search request from the frontend.

In this case, when user type for example driver query term in a frontend, the frontend will use this and send it to the search service. The search service use this query term and search the existing most real index and see getting a result, leveraging the Elasticsearch generically based on the ranking and then construct a list of result based on the priority and return to the frontend.

It's a pagination result. So user will select which one that he or she is most interesting on, and then click the result. Once it click, the tool will direct him or her into a table detail page. For example, could be a driver about certain use case.

Then inside the table GitHub page, it has all the centralized metadata. It has, for example, a table name, the high watermark of this table. Who are the owners of this table? Who are the frequent user of this table? Which stack has it generated? Which GitHub source file is associated with this table? What's the table lineage for this table?

Even connect it or integrate with Apache Superset, which is a BI or popular BI visualization tool used in Lyft to do data preview or data exploration. And you also allow you to edit the metadata. For example, we encourage the domain expert to say they have to type, for example, this column is known for this kind of purposes. That column maybe is not okay and shouldn't use this kind of power tribal knowledge.

[00:51:02] JM: I guess I'd like to revisit this question of the difference between Amundsen and the other data discovery tools that are out there. I guess Airbnb was not open source. Maybe that's a reason why you didn't use it, but there are these other industrial enterprise solutions. I know Amundsen either has been open sourced or it's going to be open sourced. So people can kind of evaluate. Now they have their own build, buy, adapt decision to make.

Do they buy one of these enterprise solutions, or do they adapt the open source Amundsen platform. How would you advice people who are looking for a data discovery platform that may perhaps considering Amundsen. How would you contrast Amundsen with the other industry leaders?

[00:51:51] MG: Yeah. So I think it's a question worthy of talking more about. So a few things that helped us make our decision, and then we can jump into a few things that can help people make their decision. For us, it was very important to have two things. One was RedShift support, and the other one was to have superset support and mode support.

So why is that? Because some of our data at that time was in RedShift and we wanted to be able to index that. Then superset and mode are two heavily used BI tools. Superset is a self-supported here at Lyft, while mode is a vendor.

When we were looking for tools, we wanted to see how well they integrate with the ecosystem we had at Lyft, and that was a fairly clear answer for us in terms of which tools we could pick. So the commercial offerings weren't integrating well.

I think we have also a strong bias to controlling our destiny through open source, and the most prominent open source tool at that time was Apache Atlas. That left a lot to be desired in terms of that experience of search as well as the detail that you get on a table page. So we felt that

the experience of Atlas was a huge gap to what we wanted it to be. That's how we ended up making this decision.

For people making the choice now, I think if you are in a place to control and adapt an open source tool, Amundsen would be a really good fit. I think it comes with a really good graph database at the end, and the search ranking that we're doing in the frequent users and other information that we get from the combination of audit logs as well as metadata is unique.

That makes Amundsen a strong contender. We are in the process of announcing open source fairly soon as well.

[00:53:42] TF: To add some detail on Mark said. So Amundsen is designed as open source from day one. So when we're building, for example, like metadata service and search service, we think of how to make it easy extendible for other company or other user to adopt.

So when we built it, we just make it as a [inaudible 00:54:02] layer in metadata services, search services, so that you could actually plug in your own – You could choose your backend entry. You don't necessarily need to be Neo4j. You could view, for example, using other graph database or even relation database or some other offering. As long as we build our abstract API proxy, so you just need to build your own proxy and implement that API. Then our frontend could immediately work in this case.

[00:54:35] JM: Okay. Well, as we begin to wrap up, I'd love to get broader perspective from both of you on data engineering across the industry. You were both at Strata. I was at Strata. It was a nice time to - I guess, pause and reflect on just the state of the industry. What are the newer aspects of data engineering that you're excited about?

[00:54:59] MG: Yeah. A few things that I noticed at Strata that was super exciting and cool were in two areas. One was around data science productivity and the other one was around data engineering productivity. Around data engineering productivity, the things that were super exciting were doing anomaly detection as the ETL is running so that the likelihood of you generating bad data from which bad decisions get made is reduced. So instead of putting the

data out there and not having enough trust, can you short circuit the ETL pipeline so that data doesn't even go there?

Around data science productivity is mostly, I guess, biased opinion around data discovery and sort of building a new wave of data discovery applications on top of metadata and then starting to build newer compliant style applications on top of the same metadata, which is a trend we're also seeing at Lyft as well.

[00:55:54] TF: To add to that, I see – definitely, there's a change in the industry to build a better metadata engine for better data discovery. I listened to a talk for Netflix about how to build. They share their approach, share their knowledge on how to build a good metadata engine for data lineage as well as metadata use. They also talk about using graph database as their backend engine.

[00:56:22] JM: Fascinating, the rise of the graph database. Tao and Mark, I want to thank you both for coming on the show. It's been really interesting talking to you. I'm looking forward to seeing the developments in Amundsen.

[00:56:31] MG: Thank you very much for having us. It's been a pleasure.

[00:56:34] TF: Thank you.

[END OF INTERVIEW]

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a line of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]