EPISODE 582

[INTRODUCTION]

[0:00:00.3] JM: Cloud computing lowered the cost and improved accessibility to tools for storing large volumes of data. In the early 2000s Hadoop caused a revolution in large-scale batch processing. Since then, companies have been building ways to store and access their data faster and more efficiently. At the same time, the sheer volume of data has increased and machine learning has given rise to methods of extracting signal from seemingly inconsequential data points.

This confluence of factors gave rise to the role of the data engineer. A data engineer defines the data pipeline and supports data scientists and machine learning engineers. Tobias Macey hosts the Data Engineering Podcast, where he covers the fast-moving world of data engineering, including databases, cloud providers and open source tools.

Tobias and I covered a range of topics in the data engineering space, we also spent significant time discussing the world of software engineering podcasting, which is niche, but it's always fun to find a kindred spirit in the software engineering podcasting world.

[SPONSOR MESSAGE]

[0:01:12.7] JM: Today's sponsor is Datadog, a monitoring and analytics platform for cloud scale infrastructure and applications. Datadog integrates seamlessly with more than 200 technologies, so you can track every layer of your complex microservice architecture all in one place. Distributed tracing an APM provide end-to-end visibility into requests wherever they go across hosts, containers and service boundaries.

With rich dashboards algorithmic alerts and collaboration tools, Datadog provides your team with the tools that they need to quickly troubleshoot and optimize modern applications. See for yourself. Start a 14-day free trial today and Datadog will send you a free t-shirt.

Go to softwareengineeringdaily.com/datadog and get your free soft t-shirt. Thanks for listening and thanks to Datadog for being a sponsor. Let's get back to the show.

[INTERVIEW]

[0:02:18.8] JM: Tobias Macey, welcome to Software Engineering Daily.

[0:02:20.7] TM: Thanks, Jeff. Thanks for having me.

[0:02:22.5] JM: You're the host of the Data Engineering Podcast, as well as the podcast about Python which is called podcast.init. I want to focus most of our conversation around data engineering. I've done some shows recently about it. Data engineering started with the Hadoop ecosystem, arguably. Unless you go back to pre-Hadoop days, but I think most people started talking about this term around the Hadoop times.

Then it was followed by these open source tools, like Storm and Kafka, and then more recently things like Spark and Flink, and then there's a ton of proprietary tools. Describe the timeline for data engineering and how it got to its present-day state from your point of view.

[0:03:11.0] TM: Yeah, so the origination of data engineering, I guess if you want to go far enough back was really with the database administrators, because they were the who were responsible for making sure that your data was available and backed up and had an appropriate shape and schema for the application to be able to access it.

In terms of the modern view of what data engineering happens to be, the next iteration was in terms of the business intelligence suite of people being able to pull data out of application systems, and then modify it and analyze it for being able to create usable reports for the c-suite, or people higher up in the business to be able to understand how things were operating. Then as you mentioned, Hadoop came in and allowed for much more massive scale of data processing.

Then when that started to become too slow, a lot of the streaming systems came in. Throughout all of this there's been the so called ETL, or extract, transform and load paradigm with a number

of different tools available for that with some of the more recent notable additions being things like Airflow and Luigi, or Oozie for the Hadoop ecosystem.

[0:04:22.9] JM: There are this set of trends that led to data engineering becoming its own role, but in the past I think those data engineering tasks were subsumed into roles like DBA, or I guess software engineer. Maybe even data analyst perhaps, but data engineer was a newer role. You only started to see that in the last two to five years. Why did data engineering become a distinct job title? Was there so much work to be done that we had to allocate specific job roles to the task of data engineering?

[0:05:00.5] TM: Yeah, I think part of it is as an offshoot to the growing trend in data science, where businesses were realizing that their data scientists and data analysts were spending a lot of their time just trying to gain access to and manipulate the raw data to get it to a usable state. Then they started breaking off data engineering as a distinct role, so that they could gain more value from both sets where they understood that the processes of actually gathering and collecting and storing and shaping the data was complex enough to merit its own responsibility and that would free up the data scientists to focus on what they do well, where they're not necessarily experts in distributed systems, or systems administration and getting things deployed.

They're more interested in just digging into the data and using their own tools for analyzing it. Whereas, the data engineer is responsible for things like collecting the data, whether it be from event system such as clickstream data, or extracting it from application databases, and then transforming it to be then load into a data warehouse, or using things like Kafka, or Flink, or Storm for being able to do on-the-fly transformations and some coarse-grained analysis to get it to where the data scientists are able to take advantage of it and really extract the value from it.

[0:06:22.5] JM: In your first response, you addressed some of the different classes of data engineering tools, so you talked about of course the databases where we used to throw all of the data and then eventually, we got to these distributed file systems like HDFS, or arguably S3. Those are places where you can store indiscriminate types of data and in any file type. It doesn't just have to be in the in the format of whatever is your core database.

Then we've got these processing systems that developed, like Spark and Flink. Then there's also these workflow orchestration tools, like the air flows of the world, and then I think there's proprietary ones. Google has something –

[0:07:10.4] TM: Cloud dataflow.

[0:07:11.5] JM: Cloud dataflow. Right. Take me through the different classes of data engineering tools and what is the size of a company that needs to have all of these things in place and what are the things that are more rudimentary and might only be used by smaller companies?

[0:07:29.7] TM: Yeah. I mean, that's a massive space to discuss, but to break it down at a coarse granularity, as you mentioned there databases which can take the form of either an application database, which is usually referred to as an OLTP, or Online Transaction Processing, which is things like your e-commerce store being able to accept orders, or a user facing site, like a forum; the database would store just the user comments and interactions and maybe user profiles.

Then there's also things like an OLAP, or Online Analytic Processing database, which is generally architected a little differently in terms of the schema, sometimes you might use a different database engine and that's more tuned for being able to do interactive analysis. It might have some data that's pre-aggregated, and that would be used for something like a business, intelligence dashboard where you can quickly dig into certain reports.

There is things like the data warehouse, which is a larger volume of data. Much more care and effort is put into determining what the schema is going to be, because that generally determines how capable or effective you are at being able to actually run any analysis on the database, because of the volume. That's also usually the destination of an ETL pipeline. That brings us to another class of tool, which is the extract, transform and load, which historically may have been things like the sequel server integration studios from Microsoft, or there were tools like Pentaho, or Jaspersoft that had a lot more GUI-driven click and point get the data from here, run this transformation on it, dump it into here.

Now with the explosion of open source tools and different points of integration that you might need to do, or enriching the data, then you've got more flexible tools such as Airflow or Luigi or Oozie. I'm sure there are huge classes of them that I'm leaving out here, but as far as other tooling, there are things that we've – streaming that we mentioned with Spark and Samza and Flink and Kafka, and that's more for being able to do high-volume data, where you're generally just piping it from one place to another doing minimal transformations. That's for being able to do things like event sourcing, or command query responsibility segregation, stuff like that.

[0:09:51.2] JM: From all of these different fields, obviously data engineering has so much information that you could potentially explore. What are the areas that you're trying to focus on within the data engineering podcast?

[0:10:05.5] TM: Really I'm trying to explore the breadth of data management as a whole. Data engineering is it's generally defined as in some cases just a subset of that, because it's focusing more on these large-scale systems. Even for a smaller company, you might use just something like Redash or Metabase, where you just have a tool they can tap into all your existing data sources and extract data directly from there to generate some rough dashboards to get a high-level view of what's happening.

Some of the other things that I'm trying to focus on are the underlying storage systems, so one of the topics that I've been thinking about and figuring out how to put together is an episode about distributed file systems, and doing some compare and contrast between them along similar lines. One of the episodes I did a little while ago was talking about serialization formats, where I had the original authors of Avro and Parquet on to discuss their relative merits and how they fit into the overall space of data engineering and data storage.

[0:11:07.9] JM: How do those data serialization formats compare? I haven't dove into that too much, except on a few shows where we talked about Apache Arrow, I think, but that's not exactly a serialization format. What have you learned in diving into the serialization formats?

[0:11:26.3] TM: Yeah, so that episode in particular is very interesting, because Avro is more useful as an archival system and it's more row-oriented. You're able to maintain a schema of the data and evolve that schema over time. That's useful for things like long-term storage, data lake

that you might put into S3 or HDFS, and just access somewhat infrequently, or if you need to be able to access a record at a time.

Then Parquet is column-oriented, so that's more for if you're doing some active analysis on the data and you want to be able to query subsets of the fields and aggregate across all the files. Then the two formats are actually interchangeable, where they have mechanisms for being able to translate an Avro schema to a Parquet schema, so that you can for instance store all of your Avro files in S3, or some data lake and then move them into a transactional system, whether it be HDFS running Hive or using Presto on top of it for being able to do that interactive analysis across it.

[0:12:33.8] JM: You mentioned they're shuttling data from S3 to HDFS and HDFS being a place where you can have your data be more operational. I was under the impression that you could just operationalize S3 data. Aren't there a number of people who use S3 as their data lake?

[0:12:52.6] TM: Yeah, you absolutely can do that. It's just that when you're working with S3 in that capacity, you have things to keep in mind, such as the pricing for API requests, and if the systems that you're using to do the analysis live outside of Amazon, you might be paying network costs, and the latency is may be higher for if you're trying to do more transactional workflows. Whereas, if it's in HDFS, it's generally going to be slightly more i/o optimized and the systems doing the analysis might be more closely located in terms of the network topology.

[0:13:27.2] JM: Have you had a lot of conversations with people where they're trying to get their costs of their data engineering infrastructure down? Because many of the conversations I've had, it feels like there is not – cost restriction is not a big deal. The focus is more on, we've got a ton of data and we know there's value there. At any expense, we will just try to figure out how to make use of this data.

[0:13:51.8] TM: It's not something that has come up a lot in terms of the podcast episodes I've done, but I have spoken to people where they may for instance move from Redshift to Google BigQuery, because it's easier for their data scientists to do the interactions. Then they also have systems in place to be able to monitor the size and scope of the queries that are being executed, so that they can have that as a feedback to the data scientists to say, "Hey, maybe

don't run this specific query, because it's costing us a lot of money in terms of the overall data that you're scanning." You might be able to cut down costs by leaving out this column that you're not actually using further in the analysis with things like that.

[SPONSOR MESSAGE]

[0:14:37.9] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW CONTINUED]

[0:16:13.3] JM: You started this podcast about data engineering exclusively. Tell me your roadmap to just starting that. Why did you get interested enough in data engineering to start a podcast entirely about it?

[0:16:26.2] TM: I've worked as a Systems Administrator and as a software engineer and most recently as a "DevOps engineer," take that as you will. I've been interested in the data systems and the systems that are used for processing the data and the complexities around that for a while, largely because of my roles of being responsible for those systems, whether it's just managing the databases, or as a software engineer being cognizant of the way that the database schema can impact the performance of an application, or how that data can be used for other business purposes.

There are a large number of podcasts that are out there that address things like data science, and maybe focus specifically on the Hadoop ecosystem, or business intelligence systems, but there wasn't anything that I had found that addresses more broadly the topic of data engineering and the people who are doing that work. It seemed something that was interesting both personally, and it seemed a topic area that wasn't receiving the attention that it was due.

Similar to when I started podcast out in it, I saw a gap in the market for podcasts, so it was something that I wanted to listen to. Nobody else was doing it at the time, so I decided that it was time for me to take up that mantle as well. So far, I've been happy with that decision and it's been growing slowly, but steadily and I've been happy with how things have been going that far. Also, it's just an excuse for me to talk to intelligent and interesting people who are working in the space, so that I can personally learn more about it.

[0:18:04.2] JM: Yes, I can share that intention with you. Why do you think people listen to podcasts about software engineering? Because this surprises some people, when you tell them like, "Oh, yeah. I do a podcast about software engineering." They'd be like, "Why would anybody want to listen to that?"

[0:18:20.7] TM: Well, I think a lot of it is just inherent in a lot of the people who work in the industry, because they're interested in learning how to solve different problems, or just improve their skills in the space so that they can become more effective at their job, or so that they can

leverage that into a new skill area so that they can maybe start a new career, or start in a new area of a business.

Top podcasts are a great way to do that, because of the – basically just because of the nature of the medium, where as you've discussed in your show before, you can listen to a podcast when you're going for a run, or commuting, or doing the dishes, whatever it might be. Whereas, videos or textual media require much more dedicated attention. Podcasts are a good way to just keep up to date with what's going on, learn new information. If there's something that interests you, you can then take it upon yourself to dig deeper, maybe with references that are cited in the podcast, or just by doing your own research.

[0:19:17.1] JM: What's your own podcast listening experience like? How many hours per week are you listening to podcasts?

[0:19:23.8] TM: Probably too many. I was just looking at Pocketcast recently. I think I'm up to something on the order of about 80 different shows that I subscribe to. Most of them I listen to about every episode, though I do it at 3x speed so that I can actually get through them and not [inaudible 0:19:38.3] –

[0:19:38.4] JM: You're one of those people.

[0:19:40.7] TM: Yeah, I am one of those people. Yeah.

[0:19:42.5] JM: Does it ever feel unhealthy to listen to that much podcasting information? Because obviously if you told people you watch that much Netflix, they would be concerned for your health, but podcasts it's like it has more of an intellectual air to it, at least these days.

[0:20:00.1] TM: Right, and also again with the nature of visual media, it's much more all-consuming. There's not really anything else you can be doing with that time. Whereas with podcasts, you can still be productive and occasionally I'll even put on a podcast and listen to it while I'm doing work for my day job, or some of my consulting that I do on the side. Not all the time, because occasionally it just gets to be too much flowing at one time, so I'll have to pause it. Mostly, I listen to it when I'm commuting, or when I get up in the morning and I'm starting to get ready for work, stuff like that.

I try to keep a healthy of topic areas too, so I've got a lot of tech podcasts, but then I also listen to things like LeVar Burton Reads for just some casual entertainment. I listen to some podcasts about economics and general news, so I try to keep up to date with a broad range of topics, though technology is definitely very heavy-focused within that list of shows that I subscribe to.

[0:20:53.8] JM: Have you become a more isolated person as more podcast content has become available to you?

[0:21:00.4] TM: I don't think so. I mean, to be honest. I've been fairly isolated for a while. I have a family that I spend a lot of time with. I have a full-time job that I do. I've got the podcasts that keep me busy. and then I also do consulting on the side as a part-time engagement just as another way of learning new things and getting some extra income.

[0:21:21.0] JM: Right. Yeah, I guess maybe –I don't mean to project my own podcasting experience, but I did a show a while ago with the guy who does the Y Combinator Podcast, Craig Cannon. He was the first person to really bring this to my attention, because he had been thinking about it for a while, but he just has found himself becoming more introverted and more isolated the more he listens to podcast, and he's actually – he's the first person I've talked to who said he had to tamp down the amount of podcasts he was consuming, because it was too much, too addictive, too much in this audio-only world where you go an entire day not interacting with anybody just listening to podcasts. It sounds like you're not a victim of such introversion.

[0:22:07.8] TM: Yeah. I think that depends too. I mean, partly on personality, but also partly on the types of podcasts you're listening to. Some of the ones that I subscribe to are related to freelancing and business development. A lot of those will have calls to action of actually going out and networking with people and talking to people.

If you have the motivation from other sources to actually go and relate to people and have some interaction beyond just consuming media, then that will help with preventing that type of extreme introversion.

[0:22:40.0] JM: The space of motivational small business type of podcasts, I found that space to be super useful before I started Software Engineering Daily, just in terms of getting into the mindset of thinking about small business, or entrepreneurship; very crowded space of the small business podcasts. Are there any ones that stand out for you?

[0:23:06.6] TM: Some of the ones that I've been listening to for a while are the freelancers show from devchat.tv network. I recently started listening to – blanking on the names, of course. I listen to the podcasts, I know the topic areas, but I don't pay as much attention to the actual podcast name, but I can add some links to the show notes afterwards for people who are interested in seeing what my listening list and my listening habits happen to be.

[0:23:29.8] JM: Yeah, sure. I'm sure we'll go back to data engineering in a sec. I'm always like to prod people who are deep affinity for podcasting about some of their listening habits. Does it feel the podcast ecosystem is evolving, or does it feel it's in stasis to you?

[0:23:46.6] TM: I think that for a while it was in a steady state of podcasts were mainly for people who were very tapped into that space, but they're becoming a bit more popular. It seems people are experimenting with different ways of doing it. I think there are more new entries into the market, so it's forcing people who have been in it for a while to rethink ways to stay relevant, or stay interesting.

I definitely think that there is a lot of room for improvement and then there's also some technological changes, such as Apple releasing the listening metrics for people who are listening to podcasts by Apple devices. There's more of a focus on figuring out how to gain usable metrics in terms of listening audience with people building dedicated apps, or maybe people who are creating calls to action in their podcasts to get some feedback from the users. There's definitely a lot of room for new innovation and improvement in the podcast ecosystem as a whole.

[0:24:51.0] JM: How do you explain technical concepts in this format?

[0:24:55.8] TM: It's definitely not always an easy thing. A lot of times what I'll try to do is focus on the architectural aspects of projects for people that I'm interviewing and discuss some of the

ways that it has evolved and the challenges that they faced that have forced those different evolutions, or new approaches for tackling some of these problems. I don't necessarily want to dig down into the API level of you make this call to get this reaction. I try to keep it broad strokes about what is this technology useful for, what are the challenges that you've faced both technical and social, because a lot of times the hardest problems in a software project is marketing at getting people aware of it, maybe issues with people who have a negative reaction to it. A lot of times, the hardest problems in technology are the people and not the software.

[0:25:47.7] JM: Okay, so different companies move at different speeds. There are many companies that they set up Hadoop and they've got HDFS. Maybe they have some nightly hive jobs, and then they gradually build infrastructure to make their batch jobs shorter and shorter. I think over time more and more of these companies have transferred to a place where they can have some shorter latency jobs in addition to these overnight batch jobs.

On the far end of the quickly developing company spectrum, you have companies that are well-developed in their real-time infrastructure. They have data that makes it very quickly from the ingest point to the operationalized, materialized viewpoint. There are a lot of companies that are still earlier in that transitionary phase to getting their data operationalized. Do you have a perspective for the roadmap that companies typically take from that migration from a totally batch Hadoop HDFS infrastructure to a more real-time, if we're talking about the extreme end of the super modern infrastructure? Is there a tried-and-true roadmap, or is it just like janky and everybody's got their own thing that they do?

[0:27:15.7] TM: It's definitely very context-dependent, where some companies may not even have the infrastructure for setting up Hadoop, where they’re may be just using Redash, or Metabase, or something similar to gain and that need to do some rudimentary analysis directly against their application databases. Then they might add some ETL jobs to pull that data out and put it into a dedicated database, whether it's just another PostgresSQL instance, or something more purpose-built like Redshift or Snowflake.

Then if somebody already has the Hadoop infrastructure, they might use something like Apache drill to just put a sequel interface on top of the data, so that you can get more transactional analysis out of it. Then if you're talking about event stream data, people might decide to set up

Spark, or Kafka, or they might go with some of the increasingly available hosted solutions, whether it's confluent with Kafka, or Databricks with Spark. That can be an easy on-ramp to the big data world, where you don't necessarily have to have the in-house capacity to set up and manage these systems, which can definitely be very complex to get set up and get right.

With the fact that there are so many companies being built around some of these solutions and companies such as Astronomer, or Stitch that will handle your ETL for you, it becomes a lot more accessible for smaller to mid-sized companies to actually be able to leverage their data more effectively.

[0:28:54.7] JM: When you say small, how small are we talk – what's the smallest company with a significant data engineering pipeline that you've seen?
[0:29:02.8] TM: I mean, generally a lot of small companies, like I said will use these hosted solutions. I don't have any specific cases in mind for small businesses with a significant presence in the data engineering space. Speaking personally, I'm actually currently in the phase of we've got Redash setup, tapped into our application databases, we've got read replicas so that we don't lock-up production transactions.

We're starting to think about setting up the ETL workflow for being able to build up a data warehouse, but I think one of the best pieces of advice for people who have event or transactional data that they want to be able to capture is to just very quickly establish a schema around it and then even just dump it into S3 or Google file storage, so that you have the data available for when you do build up a more robust solution, you can then transfer it into your data warehouse, or your live analytical systems.

[0:30:05.3] JM: I know in the earliest days of the data engineering cloud products, you would have a company like AWS, or Cloudera that would take these open-source projects and they would productize it in a pretty straightforward fashion. The Clouderas and the MapRs and the Hortonworks of the world did a great job of building these companies that would come in and make Hadoop work on your infrastructure.

AWS came out with, I think the elastic MapReduce stuff, which was from my point of view a more self-serve model of the Hadoop installation that these companies would do. Then more

recently you have companies unveiling these products that they built from scratch, so you have – I mean, I guess AWS has been doing this for a while. Kinesis is probably what? Eight or 10 years old, something like that, maybe seven years old.

[0:31:05.0] TM: It has been for a while.

[0:31:06.7] JM: You have, obviously Google's got products Dataflow, Azure has cosmosDB, so companies are building stuff from scratch now. It's not just the open source solutions. Are there any notable data engineering products from cloud providers you've seen recently where there's just no parallel in the open source community?

[0:31:30.6] TM: I think the only one that really stands out significantly to me right now is the glue project from Amazon. I mean, they're definitely open source systems that are capable of doing similar things, but the –

[0:31:42.6] JM: What is Glue?

[0:31:43.6] TM: Glue from Amazon is a metadata store and provides a lot of automation around your ETL workflows, where it does schema introspection for your source data, and then it will provide the transformation steps for instance, for taking some JSON data from S3 and loading it into your Redshift cluster, or being able to take data flowing through Spark and move it into another destination system. It provides a lot of the fine detail work for you, so it relieves some of the overall workload on your data engineering team for tracking and codifying some of that information.

It's a fairly recent product. I think they just announced it at their last re-invent, but it's a pretty interesting product that I think might motivate some people to at least partially move on to Amazon, to be able to leverage it.

[0:32:36.8] JM: We did this series of shows recently around streaming infrastructure. One of the big questions I had at the beginning was that I didn't really understand what streaming meant, so there are a lot of contexts that the word data streaming is mentioned in the data engineering

world. Can you explain streaming to me like I'm five-years-old? Explain some of the different ways that the word streaming gets used?

[0:33:06.0] TM: Yeah, so as with so many things in tech, it's a very overloaded term. Originally it was used as a juxtaposition of the batch workflows that was popularized by MapReduce on top of Hadoop. Some of the early entrance to that were Storm and Spark. Then that there are different gradations of streaming too where Spark technically has what are called micro batches, where every N number of seconds, it will process a batch of data.

By all intents and purposes, it's streaming the data through. Whereas, things like Flink are more true streaming, where every single event gets processed through the pipeline. Then there are things like Kafka and Pulsar, where they're more of a durable cue, so you can stream the data into them and out of them, but they're not generally doing any actual processing on top of that data. Spark and Storm and Samza and [inaudible 0:34:06.2], those are all pipelines for being able to ingest data, maybe do some transformations, or do some machine learning algorithms on top of the data and then pipe that data out to another system, or maybe consume it directly from the spark or the storm pipeline.

Whereas Kafka and Pulsar are as I said, just a cue where you put the data in and then it sits there until you take it out so those are useful for being able to do things like fan-in or fan-out topologies, where one producer is putting in data and then you have multiple systems that are consuming from it, or if you want to do the event-based application architecture, where all of your transactions end up in a Kafka topic and then you have a consumer that will replay them and then that ends up being the final record in your database for instance. Then you can also take those same events and travel back in time to see what the state of your information was at any given point.

[0:35:03.8] JM: Pulsar, that's an Apache project?

[0:35:06.1] TM: It is an Apache project. It originated at Yahoo and it's somewhat of a competitor to Kafka. They've actually got an API compatibility layer, so that you can drop it in wherever you're using Kafka, but it's got a bit more flexibility in terms of the ways that you can use their cueing. Then also the storage layer is separated from the queue processing layer, so it uses the

Apache bookkeeper project underneath. There's a bit more granularity in terms of how you define the durability and the high availability of the system.

[0:35:38.2] JM: Okay, well you outlined the thing that was always confusing to me for a while, which was that, okay gap streaming, or your streaming data from point A to point B over a network. Then you can have the abstraction of a stream that is sitting in Kafka for example, where you have an append-only queue of data points, or files perhaps. Is that an accurate distinction to draw there between these network point-to-point communications where streaming is occurring, versus the abstraction of a stream that you operate over?

[0:36:17.3] TM: Yeah, I think that's a good way to break it down. The network stream, that can be absolutely anything. That can be a video that you're watching on Netflix that's streaming over the network. You can have the data stream going into your Kafka or Pulsar queue, you can have your streaming analysis that's happening in Spark or Samza, there's also over the past couple years the advent of streaming SQL with things like pipelineDB, where you can define a SQL query, and then as new data gets written into the database, that SQL query is continually updated so that you can actually have that streaming query fed into another system, so it's a different way of doing your continuous analysis of a data point, so you're not necessarily running a complex machine learning algorithm on it, but you're enriching new data with existing references from another table for instance so. Yeah, there are a lot of gradations of streaming and overlaps in terms of some of the usability of those systems.

[SPONSOR MESSAGE]

[0:37:25.2] JM: The octopus, a sea creature known for its intelligence and flexibility. Octopus Deploy, a friendly deployment automation tool for deploying applications like .NET apps, Java apps and more. Ask any developer and they'll tell you that it's never fun pushing code at 5 p.m. on a Friday and then crossing your fingers hoping for the best. We've all been there. We've all done that. That's where Octopus Deploy comes into the picture.

Octopus Deploy is a friendly deployment automation tool taking over where your build or CI server ends. Use Octopus to promote releases on prem or to the cloud. Octopus integrates with your existing build pipeline, TFS and VSTS, Bamboo, Team City and Jenkins. It integrates with

AWS, Azure and on-prem environments. You can reliably and repeatedly deploy your .NET and Java apps and more. If you can package it, Octopus can deploy it.

It's quick and easy to install and you can just go to octopus.com to trial Octopus free for 45 days. That's octopus.com, O-C-T-O-P-U-S.com.

[INTERVIEW CONTINUED]

[0:38:56.6] JM: When you talk to these data engineering providers or companies that are building data engineering infrastructure, what are their concerns around streaming? Because I've asked the questions of how do Spark and Flink and Kafka streams trade-off against one another?

There are the trade-offs of the durability and the latency, but I think also there's the relevant trade-offs around what are the programming paradigms that you're allowed to use in these contexts? Do you want to be using standing SQL queries like pipelineDB? Maybe you want to write your application in a way where you have standing SQL queries, then maybe you don't want to have to go in and write a complex Java application in order to support that same functionality, which you might have to do if you were programming in Spark.

[0:39:52.1] TM: Yeah. I mean, there are so many different contexts in which these systems get used and abused. One of the things that's interesting is that the Python language has actually started to become a common platform for being able to interface with these systems, although there are of course trade-offs there in terms of having to translate the data from one format to another as it goes from being a Python object to a Java object in the case where you're interfacing with spark, but then tools like Apache Arrow are becoming the universal representation of the data. so that it's an in-memory way of translating between different languages and runtimes.

I think that there is a lot of innovation happening in terms of different streaming systems, but there is also a lot of effort being put into making some of these systems more easily interoperable and interchangeable. There's actually a specification that I saw briefly, I haven't looked deeply into it, that's called open streaming that's trying to create a unified data format, or

API for making it easier to make all these different systems work together, so that it's easier for people who are trying to interoperate with them both at the source and the destination level.

[0:41:08.8] JM: Isn't that what Apache Beam does?

[0:41:11.3] TM: Yeah. Beam has become one of the unifying layers of multiple streaming systems, where the Beam engine can integrate with different systems, but I think the open streaming standard that they're trying to create is more along the lines of making it so that Kafka and Spark, etcetera, are able to interoperate better.

Again, there are multiple competitors in every space and standards processes come about from multiple places. As the old XKCD comic goes, there are 99 standards for this one thing. I'll go ahead and create another one situation, now there are 100 competing standards.

[0:41:50.7] JM: Right. The Beam conversation, Apache Beam and Google Dataflow and Apache Beam are too closely related projects. Dataflow, I find an interesting case because you hear tons of adoption of Kubernetes, you hear tons of adoption of Tensorflow, you hear tons of adoption of BigQuery. You don't hear as much about Google Cloud Dataflow, and I'm sure people are using it and I'm sure it's a great project.

It's just I wonder, because the other Google cloud services seem completely beloved and rapidly adopted, but in terms of the streaming infrastructure, it seems like Spark and Flink have a little bit more traction than Dataflow. Do you have a sense of why that is?

[0:42:34.5] TM: Well, Dataflow is actually more about defining the workflow, so it's actually a bit more analogous to Airflow, or Luigi. Whereas the streaming data pipeline in Google –

[0:42:43.4] JM: Oh, this is what I was missing.

[0:42:45.2] TM: Yeah, so the streaming data pipeline in Google was actually the Google cloud pub/sub, which is just a public subscribe system, which is along the lines of Kafka, or Kinesis.

[0:42:54.9] JM: Oh, my God. This is what I've misunderstood the whole time. Dataflow has nothing to do with the Spark, or the Flink side of things. Is there just not really a Spark-Flink type of competitor in the Google ecosystem and the Google suite of technologies? They just have workflow managers and a pub/sub system?

[0:43:16.2] TM: Yeah. I mean, I'm not as familiar with the complete suite of Google projects. but yeah, I think the intent is that basically you would use the Dataflow with the pub/sub for being able to tap into the publish/subscribe system, perform an operation on it and then submit it back to another publish, so you can get something analogous to a Spark system that way. Spark has become the elephant in the room for streaming analytics, so I think Google is relying on leveraging that ecosystem without trying to necessarily compete with it as much, because it's so well-developed and well-established as the somewhat de facto standard.

[0:43:56.8] JM: How does somebody architect something in Dataflow? Do you need to use Spark, or Flink together with the Dataflow, or do you just write Python programs and you can schedule them using Dataflow?

[0:44:11.3] TM: Dataflow is actually its own execution engine and it has, I know Python and a Java API, they may have added other ones since the last time I looked. Yeah, you would use dataflow in conjunction with some of the other Google resources. Because it's all within Google's data centers, there's the fabulous network bandwidth and low latency, so that's one of the reasons for staying within the Google ecosystem.

[0:44:33.7] JM: Interesting. I went to Strata recently and I was trying to see the world through the eyes of these different data companies, namely Databricks and Confluent, because these companies have a lot of momentum behind them. They're trying to figure out a narrative, I think. Like Databricks and Confluent seem to have somewhat different versions of an ideal stack. Of course, in the Databricks version, there's more centralization in Spark. In the Confluent version, there's more centralization in Kafka. Do you have an idea of how these different visions contrast with each other?

[0:45:15.9] TM: I think it really just plays into what you're trying to do with your data in motion, where if you're trying to perform active analyses on the data as its flowing through, then Spark

and Databricks would be more useful to you. Whereas, if you want a persistent record of events as they happen and a scalable way of being able to ingest and process that data, then Kafka is more the solution that would fit your workflow.

[0:45:45.9] JM: I know you haven't been covering machine learning too much, but you can't have a data engineering podcast without having some concept of how machine learning works, or at least how the data ingest for machine learning works and how it's used on after it's processed. How are you approaching machine learning in your podcast, as well as from a personal standpoint, are you trying to teach yourself different aspects of machine learning?

[0:46:12.6] TM: Yeah, I'm definitely trying to keep up to date, at least at a high-level with some of the trends and machine learning. A lot of the more ML and AI-focused topic areas I end up covering in podcast.init, because there are a lot of Python tools that play into that. Most recently, I released an episode about a project called polyaxon, which is a layer on top of Kubernetes for being able to scale deep learning training algorithms, so it's in some sense the CI and CD system for deep learning, where it has compatibility with Tensorflow and Keras and PI torch and things like that.

Then from the data engineering space, I try not to dig too heavily into the machine learning and heavy analysis aspects of it, because of the fact that there are so many other podcasts that cover that. It's difficult though, because there's a very vague boundary between what is data engineering, versus what is data science. Now it's becoming even fuzzier with the recent discussions around the machine learning engineer position.

It's always difficult to figure out where do I draw the line of topics that I want to cover, and I also try not to focus too heavily on one subset of the space, so I don't necessarily want to do 10 episodes in a row about different databases, because well they're definitely very critical to data engineering, they're also not the entire picture. It's always difficult figuring out what the balance is for topics as you try to keep on top of how things are evolving within the space.

[0:47:46.7] JM: Not to mention, the struggle I have is if you go very deep on a topic, you can develop some proficiency in it, like I just did 25 or 30 shows on blockchain stuff. By the end of it, I started to feel comfortable in having conversations around these topics. Machine learning is

not something I've done that with, data engineering is not something I've done that with. I still feel pretty out of my comfort zone talking about these topics.

There's this tradeoff where if you go deep, then you can develop a good repertoire and you can know the vocabulary, you can know the concepts to discuss, you can talk about the tradeoffs with some degree of fluency, but then your listeners will be like, “Why are you still doing shows about serialization formats? I'm tired of this.”

There's a trade-off between you're either developing a proficiency in something and doing too many shows on it, or you're a mile-wide and an inch thick on a wide variety of topics. I don't know, it's a trade-off I haven't quite figured out myself.

[0:48:50.7] TM: Yeah, it's eased a little bit for me, because of the fact that the two podcasts that I do are one, they're not as broad as yours as where you're trying to encompass the entirety of software engineering, and I'm focusing just on the subsets of data engineering and Python, which are both subject areas that I work fairly closely with for my day job and in my consulting. I gain the general understanding of the area, just by virtue of working with it all the time, and keeping up to date with what's going on in those areas. Though, there are definitely a lot of cases particularly with podcast.init where I might be interviewing somebody who has written a Python library for doing experimentation for doing psychology experiments, where I'm obviously not terribly adept at the ins and outs of psychology, but I at least have the focal area of Python to be able to branch out from.

It's definitely a challenge to try and figure out, is this a show that I can do proper justice with this topic area, and how much research do I need to do ahead of time to be able to speak intelligently about it and craft useful and informative questions about the space, which I have varying levels of success with, where some episodes I might come out of it saying, “I didn't do nearly enough research ahead of time to be able to properly enumerate the space that this tool is trying to solve, for instance.” It's an ongoing battle, sometimes I'm more successful than others.

[0:50:24.0] JM: Indeed. Your background; you did physics in undergrad and then eventually you went back for computer engineering. I think your second degree, you got mostly through self-

directed online study, but you had some direction because it was – you did some online learning.

When I went to college, I was so bad at it. I was so undisciplined. I didn't really learn as well as I probably could today. I sometimes think about how nice it would be to go back and just crush it with a manor of discipline that I've learned over time. It sounds that was what you were able to do, but maybe you haven't had success with your physics degree. I know there's a lot of people who are listening to this, and they're retraining from a different field, or they're retraining from – they're going back to school to study computer science, or to study software.

They're trying to reinvent themselves. I'm wondering if that was how you felt about your return to school for computer engineering, if it was a self-reinvention, or if it was just an evolution from the same things you were pursuing in physics. If you have any advice for people who are going through that reinvention process.

[0:51:42.0] TM: Sure. Yeah, so when I first got out of high school, I did with all the teachers and guidance counselors tell you to do, just go straight to college. I was very interested in science at the time and still am, and so I decided, “Oh, I'm going to go and get my PhD in theoretical physics, because that's fascinating stuff.” Got into college and realized that one, that wasn't really a viable career direction. Two, I just didn't have the focus and motivation to do a proper justice.

I actually ended up leaving school for a while and just got a job. After working at just run-of-the-mill manual labor job for a couple years, I realized that I needed to go back to school for something that would actually provide a decent living. I've always been very interested in how things work and how they're put together, which is why I was interested in doing physics in the first place. I also grew up doing construction and I've always been fairly capable with computers, though I hadn't really ever done any proper software development, or software engineering.

As I was looking around and researching what to get my degree in, I settled on computer engineering, because it's somewhat of a hybrid between electrical engineering and computer science, and so it covers that nice middle ground where you’re able to comprehend the problem from both ends. I feel it's taught me a lot of the necessary fundamentals to have appropriate

mechanical sympathy, so that what I'm working on automating cloud architecture, or writing software, I have at least a tangential understanding of what's happening at the physical layer to be able to write my programs in a way that it doesn't needlessly stress that infrastructure.

As you pointed out, when I went back for my degree in computer engineering, I did it online. At the time, I already had a fulltime job, I had a family. I did it nights and weekends with a lot more focus and dedication. I was able to get a lot more value out of it and I actually ended up starting as a Systems Administrator after about a year into the degree.

My schooling and my day job were able to play nicely off of each other in terms of what I was learning in both places. I feel that was one of the really key elements to my success in my degree program and in my subsequent career as a software engineer and as a systems administrator and somebody working in the tech industry.

[0:54:15.0] JM: Oh, yeah. Well, that apprenticeship phase of either having internship while you're in college, or you did it having a copacetic combination of going to school either online, or in person together with a fulltime job where you're actually applying some of the skills that you're learning. The the feedback loop between those two domains can be really productive and satisfactory, because you're not throwing away money on a degree while you're not making money from that knowledge, and you're not going to a job where you don't feel yourself leveling up as much, because even if even if you were, you would be learning on the side. It's the education plus occupation path is pretty rewarding.

[0:55:06.7] TM: Yeah, absolutely. That's also part of why I continue to do freelance consulting, is because it provides another outlet for learning new things and being able to play my day job off of my consulting. I might take on a consulting engagement where I'm working with Kubernetes, where that's a system that I'm working on evaluating for my day job. There are a lot of potential synergies between the two activities.

[0:55:36.0] JM: All right. Well, Tobias Macey, I will continue to listen to the Data Engineering Podcast and anybody who enjoys the topics that I have covered in a skin-deep manner; Kafka, or Spark, or these other systems that Tobias has dove into a little more deeply, I recommend checking out the Data Engineering Podcast. Thanks for coming on the show.

[0:55:57.8] TM: Thanks for having me.

[END OF INTERVIEW]

[0:56:02.0] JM: Apps today are built on a wide range of back ends, from traditional databases like PostgresSQL, to MongoDB and Elasticsearch, to file systems like S3. When it comes to analytics, the diversity and scale of these formats makes delivering data science and BI workloads very challenging. Building data pipelines seems like a never-ending job, as each new analytical tool requires designing from scratch.

There's a new open source project called Dremio that is designed to simplify analytics on all these sources. It's also designed to handle some of the hard work, like scaling performance of analytical jobs. Dremio is the team behind Apache Arrow, a new standard for end-memory columnar data analytics.

Arrow has been adapted across dozens of projects, like Pandas, to improve the performance of analytical workloads on CPUs and GPUs. It's free and open source. It's designed for everyone from your laptop, to clusters of over 1,000 nodes.

Check out Dremio today at dremio.com/sedaily. Dremio solved hard engineering problems to build their platform, and you can hear about how it works under the hood by checking out our interviews with Dremio's CTO Jacques Nadeau, as well as the CEO Tomer Shiran. At dremio.com/sedaily you can find all the necessary resources to get started with Dremio for free.

I'm really excited about Dremio. The shows we did about it were really technical and really interesting. If you like those episodes, or you like Dremio itself, be sure to tweet @dremiohq and let them know you heard about it from Software Engineering Daily.

Thanks again to Dremio and check it out at dremio.com/sedaily to learn more.

[END]