EPISODE 469

[INTRODUCTION]

[0:00:00.3] JM: Thumbtack is a marketplace for real-world services. On Thumbtack, people get their house painted, their dog walked and their furniture assembled. With 40,000 daily marketplace transactions, the company handles significant traffic.

In yesterday's episode, we explored how one aspect of Thumbtack's marketplace recently changed, going from asynchronous matching to synchronous instant matching. In this episode we zoom out to the larger architecture of Thumbtack and how the company has grown through its adoption of managed services from both AWS and Google cloud.

The word serverless has a few definitions. In the context of today's episode, serverless is all about managed services, like Google BigQuery, Google Cloud Pub/Sub and Amazon ECS. The majority of infrastructure at Thumbtack is built using services that automatically scale up and down. Application deployment, data engineering, queuing and databases are almost entirely handled by cloud providers at Thumbtack. For the most part thumbtack is a serverless company, and it makes sense.

If you're building a high-volume marketplace, you're not in the business of keeping servers running, you're in the business of improving your matching algorithms, your user experience and your overall architecture. Paying for lots of managed services is more expensive than running virtual machines, but Thumbtack saves money from not having to hire site reliability engineers

Nate Kupp leads the technical infrastructure team at Thumbtack, and we met at QCon in San Francisco to talk about how to architect a modern marketplace. This was my third time attending QCon, and as always, I was impressed by the quality of presentations and conversations that I had at QCon. They were also kind enough to set up some dedicated space for podcasters like myself.

The most widely used cloud provider is Amazon Web Services, but more and more companies that come on the show are starting to use some of the managed services from Google. The great news for developers is that integration between these managed services is oftentimes pretty easy as just an API call. At Thumbtack, the production infrastructure at AWS serves user requests.

The log of transactions that occurs across that production infrastructure gets pushed from AWS to Google cloud where the data engineering occurs. On Google cloud, the transaction records are queue in Google Cloud Pub/Sub, which is the message queuing service and those transactions get pulled off of the queue and stored in BigQuery, which is a system for storage and querying of high volumes of data.

BigQuery is used as the data lake to pull from when orchestrating machine learning jobs. These machine learning jobs are run in Cloud Dataproc, which is a managed service that runs Apache Spark. After training a model in Google cloud, that model is deployed on the AWS side where it serves user traffic.

On the Google cloud side, the orchestration of these different managed services is done by Apache Airflow, an open-source tool that is one of the few pieces of infrastructure that Thumbtack does have to manage themselves on Google cloud. It's kind of funny that this Apache Airflow workflow management tool is managing a bunch of managed services, but Airflow itself is not managed.

If you want to find out more about the Thumbtack infrastructure, you can see some great diagrams and check out the video of the talk that Nate gave at QCon. That should be on YouTube shortly if it's not already at the time of publication of this episode. You can also check out the Thumbtack engineering blog, which I have linked to in the show notes for this episode.

Thanks again to the team at QCn for allowing me to show up and do some interviews and do some research there. I really enjoyed the QCon Conference.

[SPONSOR MESSAGE]

[0:04:09.6] JM: Women 2.0 is a company with a vision of gender equality in the tech world. Women 2.0 is a community, a media company and a jobs platform that connects top female talent with engineering jobs around the world.

At the new Women 2.0 jobs platform, find a vetted jobs for women engineers, data scientists and product managers. To find a job that is right for you, go to women2.com/sedaily, and if you're an engineering company, you can connect with top female talent on Women 2.0.

Companies like Twitter, MongoDB and Craigslist use Women 2.0 to find new hires. Go to women2.com/se daily to find out how to post your company's jobs to Women 2.0.

Thanks to Women 2.0 for being a new sponsor of Software Engineering Daily, and check it out women2.com/sedaily.

Thanks for listening, and let's get back to the show.

[INTERVIEW]

[0:05:18.3] JM: Nate Kupp is an engineering manager with Thumbtack. Nate, welcome to Software Engineering Daily.

[0:05:22.3] NK: Awesome. Thanks.

[0:05:23.4] JM: We're here at QCon. I just watched your talk about scaling marketplaces at Thumbtack, and you gave a great description for how the company has advanced, and in particular how it has moved to many managed services. I think the idea of managed services has coincided with the term serverless, and you began your conference talk with a discussion of this is ambiguity. The fact that we have one terminology for serverless, which is the functions as a service systems and then we have this other terminology, which is the managed services side of things.

Do you want to start by just disambiguating these two terminologies and explain what that term serverless means from your point of view?

[0:06:14.9] NK: Yeah. I think it's a super interesting. I'd like to kind of group these together, and that the dimension that I care most about as an infrastructure engineer is the operational overhead of using these systems. As we've scaled, we've got a very small — We're a startup. We have a small engineering team, and we want to make sure that the things that we're resourcing are directionally aligned with where the business is going.

To me, anything that's just operational overhead is totally orthogonal to the goals of the business and not a worthwhile place for us to invest engineering time. In a lot of ways, we care a lot more about that than we care about dollars. Of course, we care about minimizing our costs, but engineering is such a scarce resource at a small startup that I think across serverless, across managed services, that's the compelling piece here.

I guess to answer your question, I think about serverless is like, "Okay. We'll draw a circle around any of these things. That's where we are." Taking that operation piece out of the picture where I'm not worrying about servers that are running and I'm not worrying about the machines.

In a lot of ways, I think Google has done a really good job of providing these kind of things in terms of BigQuery, right? I never worry about what machines are backing my BigQuery tables, and it means that I can kind of remove that from my mental model of how a system works. I worry about the APIs to put data in and I worry about people writing SQL [inaudible 0:07:46.3]. That's it.

[0:07:48.4] JM: The core competency of Thumbtack is to be a matching marketplace between people who need home services and workers who are willing to do those home services. The core competency of Thumbtack is not managing servers. Ideally, you would like to have the operational aspects of your infrastructure be taken care of by a company like Amazon or Google so that you can just think in terms of what sort of machine learning jobs do we need to get done. What kind of services? What kinds of abstractions do we want to offer to our developers so that they can be higher leverage and more productive.

Can you give an overview of the infrastructure at Thumbtack? I know your operational infrastructure is on AWS and the data infrastructure is on Google. So you've got a multi-cloud set up. Paint us a picture for how that looks.

[0:08:50.9] NK: Yeah. I like to think that anything on a synchronous path to a user is on the AWS side, and then kind of the more offline stuff is on Google. To zoom in to each of those, so on AWS, all of our production serving including our microservices, the remnants of the PHP monolith, the different kind of systems supporting that, they're all running there. We run all of our services on Dockerized containers on ECS and we use PostgreS and Dynamo along with a little bit of Elastic Search as our primary data stores over there. We have a bunch of touch points where we're kind of piping data in and out of that to the GCP side of things, primarily though Cloud Dataproc where we run Spark jobs, and then landing an awful lot of that in BigQuery which then feeds downstream analytics and data science and things like that.

[0:09:43.8] JM: You and I were talking before the show about how little operations burden there has been, how few outages there have been, and you think about building a marketplace business 5 or 10 years ago, even 3 years ago. One of the earlier shows that I did was with the Uber CTO about just the infrastructure of Uber. I'm sure Uber still has outages and craziness just because the volume of transactions are going on so insane. But I do have a sense that even just in the last two or three years there have been massive improvements in how few outages a company can get to by having their infrastructure in managed services. Can you talk a bit about that? I know because you've been at Thumbtack since 2014, have you seen that drop in outages over that time?

[0:10:38.3] NK: Yeah, and I think I'll attribute that both to this move to more managed services and awful lot of investment in our infrastructure from the engineering team. Absolutely, I think often a lot of the outages are problems that we cause ourselves, right? And a bad deploy, something went wrong, we brought down some system. The more that we can kind of raise the level of abstraction that our developers need to worry about, the less prone we are to those sort of outages and we can kind of offload that responsibility to one of the cloud providers. To me, that's pretty compelling, and I think kind of looking where the puck is going over the next few years, I think that that model of just raising the level of abstraction — Serverless is this kind of one step in that direction. I think there's a lot more that's coming, and to me it's super exciting,

because it means that all of us at small startups don't need to worry about that stuff anymore, and we can focus on the things that are going to move the business forward and not these operational issues and site outages and things like that.

[0:11:40.4] JM: On the user-facing infrastructure side on AWS, you've got people that are getting matched with workers to do their jobs. For example, you have somebody who goes to Thumbtack and they say, "I need my house painted," and their request gets matched with people who are willing to paint the house. They get matched up. The job goes through. Everything is great. That data gets written into your production infrastructure side, I think, in DynamoDB, and you have all these day-to-day transactional stuff. At some point, that data gets loaded or transferred somehow into the Google side of things where you have, like you said, Dataproc and other machine learning jobs.

Before we talk about the data infrastructure side, how does the data — What's your pipeline of transferring data from the AWS side to the Google cloud side?

[0:12:37.5] NK: Yeah. There's kind of two modes. We have the near real-time streaming infrastructure and then we have batch ETL. For like our data stores for PostgreS, for Dynamo, right now it's mostly batch. We're working on building change data capture out of PostgreS to pipe that — Piping streaming changes into the Google side.

I'd say that the primary source more real-time data, we log events out of the website, out of our services. These events are JSON or thrift objects. These hit a Fluentd cluster running on the AWS side, which then feeds those into Google Pub/Sub and then we have various things downstream on the Google side which subscribe to that data and process it from there.

[0:13:22.6] JM: Okay. You have an event stream on the AWS side and that event stream is like the change log of everything that is changing across your production infrastructure, and you said something called Fluentd is reading that and transferring and communicating that to the Google side.

[0:13:40.1] NK: Yeah. We're just using that as a relay into Pub/Sub, and it's a pretty quick hop rate into Cloud Pub/Sub, and then we're sending on the order of hundreds and millions of events per day through Pub/Sub and then going from there.

[0:13:52.7] JM: You just buffer all the events into Google Pub/Sub.

[0:13:56.2] NK: Yeah, and that corresponds to kind of — You can imagine, like every user action. Basically, any kind of action that happened within some context that we want to capture, we'll capture kind of a timestamp event around that end log right into this event stream.

[0:14:09.7] JM: On the AWS side, what are you doing for — How do those events exist on the AWS side?

[0:14:17.8] NK: You can imagine in PHP we'll just collect, basically compose a JSON object and then that gets logged into this event stream.

[0:14:26.4] JM: It's that you're using PostgreS for that you said?

[0:14:29.8] NK: It's directly from the application layer.

[0:14:32.0] JM: Okay.

[0:14:32.2] NK: Yeah, the application layer writes directly into this event stream.

[0:14:35.4] JM: Okay. The event stream is just sitting in memory on some box?

[0:14:41.5] NK: It will write directly to Fluentd. The application layer handles the action, writes to Fluentd, passes on —

[0:14:47.2] JM: What is Fluentd? Can you just explain what that is?

[0:14:48.7] NK: It's really just this relay that basically can buffer some data and then feed that to a downstream consumer. That piece is kind of minor in the pathway. In some cases we may

write directly to Pub/Sub, but then basically once the event is slammed in Pub/Sub, then Pub/Sub durably persists them in a distributed queue and then downstream consumers can subscribe to a particular event topic that has events from the website or events for a particular service.

[0:15:16.2] JM: How durable is Fluentd?

[0:15:18.5] NK: Fluentd, in some ways that's a bit of a weak spot for us. We run that ourselves on sort of machines on the AWS side. At some point we may transition everything to just write directly from the application layer to Pub/Sub.

[0:15:32.6] JM: Have you evaluated Kinesis for I guess — I guess that might get costly. If you were keeping the entire event stream in both Kinesis and —

[0:15:41.7] NK: Yeah. Basically, Kinesis the AWS version of Google Cloud Pub/Sub, so those are pretty much one-to-one, alternative being in like Kafka if you want to self-host it. We looked at all three. Again, I think the thing that we found really compelling about Pub/Sub and a lot of the services in the Google side is I don't have to worry about the knobs. Pub/Sub gives me an API to write events and an API to subscribe and consume events and I don't have to worry about any of the operational details of how Google handles that on their end. Whereas like Kinesis, I have to worry about sharding and there are some knob that are there and it's minor, but incrementally all these things add up and then the more kind of these knobs that we put across our infrastructure, the more work operationally that we have to do to keep things in a good place.

[0:16:33.9] JM: Okay. Right. You haven't had any issues with the Fluentd system. Have you had any outages that resulted from like some problem with the event stream on the —

[0:16:46.1] NK: Yeah, no. I think our — Overall, this whole like infrastructure has been pretty solid. I think the places where we've hit issues have been mostly, you know, I mentioned it in a talk a couple of cases like where BigQuery has made API changes that have broken us. We've been chatting a little bit with the Google team around how we can handle that better and get

some more API stability on their side. Pretty much everything else has been really solid and very handoff, which has been great.

[0:17:13.9] JM: Okay. Cool. All these events that are coming in, the change log across the application infrastructure goes through Fluentd, it does into Google Club Pub/Sub and then you have several different things that are reading from that event stream. What are some of the consumers of that event stream?

[0:17:33.9] NK: Yeah. The primary consumer today is a Spark Streaming job, which pulls events out of these Pub/Sub topics and then writes those down into BigQuery tables. One of the things that we've found pretty compelling about BigQuery was it has these streaming APIs where I can do streaming writes into a table and it's immediately queryable in BigQuery. That means that like from the time a user took an action in production to the time that our analysts can query it is like 30 seconds at scale with millions and millions of events. That's pretty cool, and it's helped us a ton when we have any issues on the side and we want to understand kind of at a higher level, kind of more of the business level rather than the engineering operations level of what's going on. The analyst can be a good partner to us, because they can go and look at the events and see what's happening.

[SPONSOR MESSAGE]

[0:18:36.0] JM: Simplify continuous delivery GoCD, the on-premise open-source continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and you can visualize them end-to-end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.org/sedaily, find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.org/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thanks to GoCD for being a continued sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:19:36.5] JM: We had a show about BigQuery recently and this is one of the things that we kind of discussed on that was like 5 or 10 years ago, the things that BigQuery offers you today at a pretty fast SLA. This was like you needed to do as a nightly job on Hadoop, right? Now, you can do like an aggregation, a giant aggregation or a giant query on the fly and deliver it to the user as like business logic, right?

[0:20:10.2] NK: Yeah. Which is amazing, right? I think it's pretty cool from our side, and I think being able to — We still have those nightly batch jobs around as the kind of just in case version of that. Yeah, one of the things that we appreciated a lot about the way that BigQuery is set up and this kind of between Pub/Sub and BigQuery. We built this as a Spark Streaming job and then we can just reuse the same data models that we were using in the batch job in the streaming job. We got a fair amount of code we used there from across both.

[0:20:47.2] JM: That's like the whole Dataflow vision of like, "There should be no batch. There should be no streaming. There's only this thing.

[0:20:52.6] NK: Yes. I think we're kind of moving incrementally towards — I'm kind of — We're holding our breath to see where things shake out with Dataflow versus Spark Streaming and what emerges as kind of the winner in that space. We went the Spark Streaming route in the short term given that we have lots of Spark code already and it was this easy hop to go build that. Well, comparatively easy versus rewriting everything in Dataflow. We'll see. I think there's a lot of things that Dataflow does really well, and particularly this like — Where we don't have to worry about the knobs is really compelling. I think the thing that I'd like to see is just more adoption there and more people using Dataflow, Apache Beam, where I can be convinced that we're not just getting locked in to this small query of the ecosystem.

[0:21:39.9] JM: Yeah. That was something I had done a couple of shows about last year and I didn't fully understand what was going on there. I did two shows about Dataflow/Apache Beam, and what I understood about Apache Beam is this is a way of making your Spark Streaming or Apache Flink things compatible with Dataflow, but I could never really wrap my mind around

what that means. Can you shed some light on that or do you understand what they're talking about there?

[0:22:10.3] NK: Yeah. My understanding is that Apache Beam is basically the open-source — Well, it's the open-source version of Dataflow. Google provides Dataflow as a managed offering and then Beam is API compatible with that.

I think, today, those two things largely go hand in hand and then I would hazard a guess that most Apache Beam users are going to be on Google using Dataflow.

[0:22:34.7] JM: Yeah, that what confused me about it.

[0:22:38.3] NK: Yeah, I'm kind of curious to see if that changes and if there's kind of more broad adoption than just within the GCP ecosystem. I'd love to see that before we kind of dive in head first and get locked into using it.

[0:22:52.1] JM: Right. Spark Streaming — Help me understand, what is Spark Streaming doing there and why was it the choice of technologies versus the other options in that situation?

[0:23:06.1] NK: Yeah. If you have this mental model of Spark as doing distributed processing on like an array of data. I have an array of data as the inputs, maybe multiple arrays of data and I'm joining them, merging them, doing map-reduced kind of operations over those arrays of data and then piping them downstream. Spark Streaming kind of takes the same thing and then moves it to where you have these micro-batches, and you can conceptualize like Spark Streaming as working in a similar way to Spark, but working it over kind of small batches of data. You have these windowed sets of data that are coming in.

The way that we use it is we're applying the transformation logic ETL that we're doing on our data across these small batches of events that are coming in that we're reading off of Pub/Sub processing through Spark Streaming and then syncing into BigQuery on our downstream consumers.

[0:23:57.9] JM: Okay. Right. Because you need some sort of system that will transfer the data from Pub/Sub, because Pub/Sub is basically this queue of different topics. So you've got different types of events that are sitting on this queue and you need to write those events into BigQuery, because BigQuery is a file system plus — It's a file system plus a query system, so you've just got this big layer of storage and you need some way of transferring the data from the queue into the storage layer and you're using Spark Streaming to pull that data and put it right to the correct places.

[0:24:40.2] NK: Yeah. I'll say what I appreciated about it is from our perspective, the model is I'm querying an API, because there's an API on Pub/Sub and all I'm doing is pulling my events out of that API and then I'm querying another API, the BigQuery API and putting my events over there. I'm really just kind of shuffling data between these two managed services, which is great, right? Then I can do some transformation logic in the middle if I need to, but it's more or less just a connecting pipe between these two APIs.

[0:25:09.2] JM: Yeah. On the Google Cloud side, in the talk that we just saw here at QCon, you were also talking about this Google Cloud Dataproc and Airflow. We had done a show about Airflow recently, and I understand that came out of Airbnb. It's kind of like a way of orchestrating different aspects of your data infrastructure and sequencing them in a certain way to patch together different logic of different components of your data infrastructure.

Explain what Google Cloud Dataproc is and how Airflow fits into your system.

[0:25:49.8] NK: Yeah. Cloud Dataproc is a managed Hadoop and Spark offering. Again, it's basically a set of APIs that you can call and say like, "Create a cluster with some set of resources." Say, "I want to create a cluster with a hundred workers and I want them all to have 16 cores, whatever."

You hit this API and 60 to 90 seconds later you'll have a cluster sitting there. There's another API, you say like, "Schedule this workload on that cluster," and then it will go off and run that job, Spark job, MapReduce job, whatever it might be. Of course, then you can just shut down the cluster with a third call.

What we did is we wrote an Airflow operator which kind of orchestrates that whole process to bring up a cluster, schedule a job on that cluster and then shut it down and kind of wrap that as one atomic unit, an operator that we use in Airflow and then we just put all our jobs inside of those containers. We say like, “Here’s the job. Put it inside of this container. Configure how much resources we think it needs and we can kind of adjust that as the job scales and then just throw it into Airflow along with all the other things that we have running.”

Like I said, we have kind of more than 600 of these that we run on a daily basis, bring up a cluster, run it, shut it down. As our scale goes up, we basically just — Is a configuration change just bump up the size of each of these clusters and we can do it at the job kind of granularity where we can say, “This job is either running too long or using a lot of resources, so we just give it some more resources and we’re off to the races.”

[0:27:19.4] JM: Cloud Dataproc is a thing that is managing servers in a way where you don’t actually have to spin up the servers or whatever. You’re just saying, “I’ve got this machine learning job that I want to run Cloud Dataproc. Provision the stuff that I need and take care of this job for me and hand me back the answer.”

[0:27:38.7] NK: Yup. Exactly. Which is super attractive from our perspective. Especially coming from this world where in the old days, like bringing up a Hadoop cluster was go edit a bunch of XML files, distribute them across 20 machines, wire them altogether and cross fingers and hope it all works out.

[0:27:55.2] JM: Yeah.

[0:27:58.0] NK: It’s really painful, right? Then it kind of got better as Puppet and Jeff and other tools came along and you can kind of manage all of that through those tools. Still, it’s like you have to worry about physical hardware with all of these configuration deployed across the cluster and it’s a huge pain to keep everything in shape. Particularly in the Hadoop world, you often have a lot of local disc if you’re working on jobs on HTFS, which means you often maybe have a machine with 20 or 30 drives attached to it multiplied by 10, 20, 30 machines. All of a sudden, you have hundreds of discs and basically you’re multiplying your failure risk for a hard

drive. There's always something breaking and there's always like a drive failing somewhere and it's just annoying.

Being able to take all of that out of the picture and just say, "Okay. It's an API and I don't have to worry about it," is incredible.

[0:28:53.5] JM: Yeah. Airflow here is — I'm wondering why — Airflow, you must have to manage yourself though. That is a system that is actually — That's one of the rare systems that you're actually managing yourself on a server somewhere.

[0:29:09.4] NK: Yes. For now. I think we'll probably, at some point, containerize pieces of it and deploy it on top of that side of the infrastructure. Yeah, for now, that's just — We've configured it with Puppet. It's running on some machines on AWS and then kind of orchestrating the whole thing.

[0:29:26.9] JM: Is there any operational burden to that or it pretty straightforward?

[0:29:29.9] NK: I'd say the operational burden comes from some of the bugs that are around in Airflow and we've had a few of those, but largely it's been pretty good for us.

[0:29:39.6] JM: We've been talking pretty abstractly about data just blasting from the event streams on your production infrastructure, going over to Google cloud, being pulled off of Cloud Pub/Sub, written to BigQuery. You've got BigQuery jobs that are — Actually, okay. You've got the data going into BigQuery and you have business logic BigQuery jobs. I want to ask you about some of those business logic jobs, but also we just talked about these Dataproc jobs, it's like machine learning jobs. How was the data getting from — Is BigQuery your system of record? Do you move all the data from BigQuery into a Dataproc container to do machine learning jobs there?

[0:30:23.2] NK: Yeah, that's a great question. The way that things work, imagine kind of each step along a data pipeline that we build, will read from some source, whether it's Pub/Sub or some third-party API for external data or wherever we're getting data from. We read form there.

We write to Google Cloud storage and then we call the BigQuery load APIs on that data we just wrote to Google Cloud storage.

Then the next stage of the jobs will all read from Google Cloud storage, process the data and, again, write to GCS and load into BigQuery. GCS is really our system of record, and then BigQuery kind of just mirrors what's on GCS.

[0:31:03.8] JM: GCS is like the Google S3 equivalent.

[0:31:05.9] NK: Exactly. Yeah. The reason we do it that way is it's kind of expensive to read from BigQuery. If you imagine like I'm running a Spark job and I'm processing terabytes of data, I want to do my reads out for GCS versus doing my reads out of BigQuery, because I'm paying for its $5 a terabyte out of BigQuery.

[0:31:26.2] JM: I think it's a columnar store, so it's probably harder to read for individual records.

[0:31:31.0] NK: In a lot of ways, the APIs will end up doing the read, writing to GCS anyway and then like then processing the data. It's easier to just read the mirror that we have on GCS and process from there.

[SPONSOR MESSAGE]

[0:31:49.2] JM: Do you have a product that is sold to software engineers? Are you looking to hire software engineers? Become a sponsor of Software Engineering Daily and support the show while getting your company into the ears of 24,000 developers around the world. Developers listen to Software Engineering Daily to find out about the latest strategies and tools for building software. Send me an email to find out more, jeff@softwareengineeringdaily.com.

The sponsors of Software Engineering Daily make this show possible, and I have enjoyed advertising for some of the brands that I personally love using in my software projects. If you're curious about becoming a sponsor, send me an email, or email your marketing director and tell them that they should send me an email, jeff@softwareengineeringdaily.com.

Thanks as always for listening and supporting the show, and let's get on with the show.

[INTERVIEW CONTINUED]

[0:32:51.4] JM: To make this more tangible for people, could you may be give an example — I'd like to talk maybe some examples of BigQuery, particular just a random BigQuery query that comes to mind is business logic, and then also a piece of some — One of those Dataproc jobs, just to make this a little bit more tangible for people.

[0:33:11.8] NK: Yeah. Maybe I can trace through kind of end-to-end from, say, like production data to our analytics dash-boarding, you know, key company metrics. I'd say most of product data lives in PostgreS. We have these scoop jobs that we run which run on Dataproc. Scoop still uses MapReduce, but it just runs with Dataproc job on a Dataproc cluster, reads from some PostgreS table. Imagine we have like the users table in PostgreS. We have a user's table, has 30 columns for whatever it is. We pull that data out of PostgreS via Scoop, process it through Dataproc and Scoop and then land it as Avro files on GCS and then [inaudible 0:33:52.2] to put those in BigQuery.

We may have a bunch of these and we pull a bunch of different datasets into GCS, BigQuery. Then we actually have this kind of what I call last mile ETL, where we let the analysts write SQL queries for BigQuery, which they can run to take this set of tables and transform to more consumable tables for the rest of the business.

Imagine, we have all these kind of like tables which are laid out exactly the way they are in PostgreS, right? You have all your PostgreS tables, now they're kind of mirrored into BigQuery. That's maybe not the format you want in terms of measuring KPIs for the business. That mapping from like PostgreS table schemas to entities that the business cares about and all the KPIs for those entities, the analytics team owns a lot of those definitions and we just run those automatically on a regular cadence.

Then on that last step, we have now — You can imagine a user's table, maybe separate table for pros, for customers, for requests and kind of all that key entities on Thumbtack and then all

the KPIs we care about for each of those different entities. From there it's kind of this last hop of, well, create all of our key company dashboards and read form these tables and we're off to the races.

[0:35:09.0] JM: Yeah. That example — Maybe I missed it, but did you cover something that happened in Dataproc, in the Dataproc layer?

[0:35:17.9] NK: Yeah. The Scoop jobs that pull from PostgreS run on Dataproc, and I will say we also — As I mentioned in the talk, like we also pull in data that it's Dynamo. We also pull in these events from production. Often, this kind of last mile ETL will grab data from each of those to compose these key tables for each entity in the product.

[0:35:40.5] JM: Okay. We just glossed over the production infrastructure, I guess. We just talked about this big event stream that's being created and going through Fluentd and then making its way to the Google side, AWS on one side and Google on the other side. We just talked about the event stream on the AWS side. Obviously, as these events are being created across the application architecture, the AWS side of things, you want to have databases that are subscribing to that application event stream and updating. What's on the production infrastructure side of things? What's getting updated off, or who is reading form that event stream?

[0:36:21.7] NK: The event stream is strictly over to the Google side. The way that the production serving infrastructure works, so we have the PHP monolith, which is kind of the entry point for everything. Then we have a layer of microservices behind that. Those microservices we write in Go and in Scala, primarily in Go. Three or four services are written in Scala. All of these, the website and the microservices are on Docker containers orchestrated through ECS and all of these then kind of write to either PostgreS or Dynamo, read and write from both of these data stores and then also elastic search for search applications.

[0:37:04.6] JM: Do you ever have a problem where let's say a transaction comes through the system, a user gets their house painted and it all goes through and of course it gets paid and the transaction is over. You need to update the system of record that is like the history of the user like, okay, that now there's a new transaction that they've done. You also need to update

certain search indexes on Elastic Search, and so in a number of the shows I've done, they use this event stream as the source of truth and then the user database will read from the event stream and update and the search index will also read from the invent stream and update it, because otherwise you've got to contact both the database, the user database and the search index and update both of those things. Many people use the event stream as a way to decouple the reads and the writes. You have a different solution there?

[0:37:58.6] NK: Yeah. This is exactly the problem that's keeping me awake at night.

[0:38:02.3] JM: Okay.

[0:38:02.7] NK: Yeah. Essentially, what I would like to move toward — What we are moving towards is basically PostgreS is the system of record, Dynamo in some cases, but primarily PostgreS, and then we do change data capture off of PostgreS and feed the changes out of PostgreS, which then become the canonical data stream feeding all of our work on the Google side.

The events that we're logging from the application layer are in a sense a shortcut until we finish building that, because they suffer from exactly that problem. When you're looking events from the application layer, if, say, the transaction fails from some reason, then now you have this inconsistency between the event stream and the data store and vice versa. If you fail to write events to the event stream but your PostgreS write succeeded, you get this inconsistency. The ideal world is where everything is kind of going through your data store. In our case, PostgreS, and we don't have this separate event stream.

To bridge from the world where we don't have anything to the world where everything is kind of consistent through the event stream, consistent from PostgreS, these application events we've provided the engineering team with is basically big asterisk that we are not providing guarantees on a strong consistency with that's in PostgreS. It's like it's close and kind of gives us coverage in the interim, but our goal is to move towards a world where everything is built off of that change log out of PostgreS.

[0:39:34.0] JM: Is to write ahead log, right?

[0:39:35.4] NK: Yeah. Basically, using logical replication from PostgreS, you can capture all the changes, all the transaction sequence through PostgreS and feed that into your data pipelines. Since we don't have this yet, for all of like key company metrics, everything is built off of this like batch export out of PostgreS today, and events are kind of additional signal, but they're not kind of driving any of the KPIs for the business.

[0:40:01.2] JM: Sure. In your dream infrastructure, would the write ahead log would be like writing to Kafka or something?

[0:40:09.5] NK: Essentially, imagine we have this entity that's reading out of PostgreS and then writing into Pub/Sub, so we have something sitting in the middle between the — Basically, reading this replication log from —

[0:40:19.1] JM: Google Cloud Pub/Sub.

[0:40:20.0] NK: Yeah, and then writing directly to Cloud Pub/Sub, and then we have a Spark Streaming job which consumes that log, transforms it and things go from there.

[0:40:29.0] JM: Yeah. Okay, that sounds great. Then this is the Google cloud infrastructure making its way to the production infrastructure, right?

[0:40:39.8] NK: Yes. I think we'll continue kind of moving in that direction with much of these things. I think I will say it's sort of an open question on whether we'll move any like production serving infra, and that as long as we've got this tight coupling with Dynamo in a bunch of places, any sort of migration away from that would be pretty challenging.

[0:41:04.4] JM: That is interesting. I did a show — Actually, the show that was published today was with Fiverr, the company Fiverr.

[0:41:10.7] NK: Yeah.

[0:41:12.0] JM: That company kind of has similar engineering tasks as Thumbtack, right? Fiverr is sort of like a gig economy thing for digital services. Thumbtack is for mostly physical services, I believe. One thing that he said was — You're basically talking about like how are you going to implement event sourcing. One of the things he said was they're going with Kafka for event sourcing because they figure this is like kind of the Kubernetes of the event queue architecture and they think that all of these event sourcing evolutions are going to come first to Kafka and maybe they'll come — They might be later to the game on Google Cloud Pub/Sub. Have you thought about that all?

[0:41:54.4] NK: Yeah, absolutely. I think as we're building this change data capture out of PostgreS, it's an open question on whether we will go the Pub/Sub route or standup Kafka. I think we're very hesitant to add yet another piece of infrastructure we need to operationally manage, but I think that that point is absolutely right. Kafka is the status quo for this, and there's also this kind of one asterisk today with Pub/Sub, and that Pub/Sub does not provide ordering guarantees. You're putting all these messages in and you're reading them out and there're no guarantees that the order in which you can consume them was the order in which they were put into the queue. Kafka can provide ordering within partitions, right? Then in some cases, particularly if you're trying to reconstruct the state of your PostgreS database, that can be helpful.

[0:42:49.9] JM: This has been like the wonkiest episode of Software Engineering Daily ever. I think about where to wrap it up, because this has been really interesting. You had some interesting thoughts in your talk about just some general lessons that you've learned in your time at Thumbtack. I guess you've been there for about three years at this point and the team has scaled as we were talking right before the show. You've gone from moving just past the two pizza team in terms of how many people are kind of directly reporting to you. What are some of the management and scaling lessons that you've learned, because Thumbtack is growing like a weed, really fast growing company. I'm sure it's growing in terms of employees as well as in terms of traffic. What are just some of the engineering management lessons you've learned?

[0:43:42.2] NK: Yeah. One; it's hard. It's hard to do well. I think something that just has been constantly on my mind as we've grown is like how do we protect the engineering culture while we're growing the team so fast. If we're adding so many engineers, we want to make sure that

like we're able to kind of add them to the team and not just totally derail a team culture, and I think there's a big risk. When you're adding people really, really fast to an engineering team, there's kind of this carrying capacity that you don't want to exceed. It's just hard, right? It's just really hard. Especially if you get kind of big spikes of lots of new people, it can be really hard to kind of get everyone on boarded and up-to-speed and figure out all the existing systems and processes.

I think that's something that we give a lot of thought to, even kind of top of funnel in terms of as we interview people, as we bring people to the process and into the engineering team. We want to hire people who aren't jerks and people that we'd be excited to work with and just be really thoughtful about the kind of engineering team that we're building. That, I think, has been a key priority for me and for us and I think we'll continue to be as we keep growing fast.

[0:45:00.8] JM: Yeah. Okay. Nate, this sounds like a great place to stop.

[0:45:03.8] NK: Cool.

[0:45:04.0] JM: Yeah. Thanks for coming on Software Engineering Daily. It's been great.

[0:45:06.1] NK: Awesome. Thanks for having me.

[0:45:07.2] JM: Thank you.

[END OF INTERVIEW]

ensures that your app can serve traffic even under high load. Apica Synthetic sends traffic to your website in your API endpoints from more than 80 different countries ensuring wide coverage.

Right now you can go to softwareengineeringdaily.com/apica for a webinar about the real ROI of API testing. You can also find past webinars such as how to optimize websites for fast load time. Go to softwareengineeringdaily.com/apica to find the latest webinars on load testing and lots of other topics and check out Apica system for testing, monitoring and optimization.

Thanks again to Apica for being a sponsor of Software Engineering Daily.

[END]