EPISODE 1286

[INTRODUCTION]

[0:00:00.3] JM: Uber has intense data science workloads. Those data science workloads have always been intense even from the early days. In the early days of Uber, there was not that much data infrastructure that was built. Kevin Novak, was one of the first people to work on the data science infrastructure of Uber. He joins the show today to talk about a brief history of what he did at Uber. How he started the data science stack there. How he helped other people start the data science stack there. There's some great stories about early Uber, including surge pricing. Kevin was apparently, partially responsible for surge pricing.

Our first book is coming soon. *Move Fast* is a book about how Facebook build software. It comes out July 6, and it's something we're pretty proud of. We've spent about two and a half years on this book. It's been a great exploration of how one of the most successful companies in the world build software. In the process of writing *Move Fast*, I was reinforced with regard to the idea that I want to build a software company.

I have a new idea that I'm starting to build. The difference between this company and the previous software companies that I've started is, I need to let go of some of the responsibilities of software engineering daily. We're going to be starting to transition to having more voices on software engineering daily. In the long run, I think this will be much better for the business because we'll have a deeper, more diverse voice about what the world of software entails.

If you are interested in becoming a host, please email me, jeff@softwareengineeringdaily.com. This is a paid opportunity. It's also a great opportunity for learning and access and growing your personal brand. Speaking of personal brand, we are starting the YouTube channel as well. We'll start to air choice interviews that we've done in person at a studio. These are high quality videos that we're going to be uploading to YouTube. You can subscribe to those videos at YouTube and find the Software Daily YouTube channel. Thank you for listening. Thank you for reading. I hope you check out *Move Fast* and very soon. Thanks for watching software daily.

[INTERVIEW]

[00:02:29] JM: Kevin, welcome to the show.

[00:02:33] KN: Thank you for having me. This is great.

[00:02:35] JM: You were one of the first Data Scientists to work at Uber and grapple with the scale of that company. I think of Uber and other ride sharing firms as well, other logistics companies as basically an unsolvable data engineering problem, because the scale of geospatial information, customer information that is generated on the fly, and needs to be managed and turned into other actionable data points is effectively infinite.

The optimizations that you could do are effectively infinite. In such an environment, you have to figure out some platform strategy, some sustainable platform strategy for collecting the data, working with the data, and making it serviceable to the data scientists. I'm most curious about the state of that platform when you joined, and the direction it took as the company matures.

[00:03:37] KN: Sure. Absolutely. As you probably aware, I joined mid-2011. I was their 21st employee. I think there were seven engineers, I was the seventh engineer. I inherited basically nothing from the get go. What's actually very interesting about Uber is you're right, that there is massive scale today. If you think about it from a customer transactions point of view, on a transaction volume, people are using Uber, even if you're a power user, you order of single digit times per day. It's once to work maybe once to lunch once back when so.

The actual speed of customer transaction data was pretty slow, in fact that you could basically use an off the shelf MySQL Cluster for a long time, was just basically what we will do to handle transactional records and all the analytics are doing on predicting churn and predicting re engagement. These types of things could be built on top of my sequel stack. So for a long time, just because we were constrained by data infrastructure, that was where a lot of the effort got spent on the data side of things.

The obvious exception to that is the geospatial information that you get from cars. We would have telemetry coming off these Cars at one hertz, a second location, basically everything an iPhone could share with us on speed accelerometer, all of that was being shared with Uber. So

most of our data scaling constraints came from the geospatial, I guess you'd call them geo temporal logs coming off of these cars.

For a long time, we didn't really know what to do with it. When we stumbled on pub sub early on even before it really caught on with the confluence of the world and all of that. Literally our geospatial information system was, it comes in from the cars a simple basically handler to process that information a CDN type system just to basically receive from the cars published to a Kafka stream that got sent to a flat file, and we just spooled these gios logs, and did nothing with them honestly, for the first several years but knowing they were going to be valuable.

By about maybe 2014, 2015, we really felt we developed enough engineering companies on the data infrastructure side to start tackling, mapping and traffic and all of these sorts of products we wanted to build on top of geospatial information. That was where our first real data lakes got built. We started out with your classic Hadoop system. We have a bunch of friends from Cloudera who were getting those going. Realized pretty quickly, we needed something a lot more custom.

That eventually took the form of us building projects like M three, which is our time series database, in the sense, they've gone off and created them coming on top of that. One of the big things for the analytics side of things is we stumbled on the hex notation, and you'd see it as a certain area we've in surfaces and the driver app, where we realize you can basically cover the world in hexagons that were roughly the size of a city block. You can tessellate the entire globe if you put five big Pentagon, polygons somewhere on the planet.

We kept playing with it. We're able to get four of those five Pentagon's over oceans and the fifth one is over the Western Sahara Desert. Parts of like Uber Morocco, if we ever get to that part of the world, or Uber Western Sahara might have some quirky analytics at some point. Actually got a nice one, it basically removes all of the geospatial computation, because hexagons, you can just have a rank order radius of you are, you can say there is data within the hexagon of interest.

There is the ring of hexagon surrounding it. The ring of hexagon surrounding that. So it's actually a lot more computationally simple to think of the world rather than as continuous GPS

points. It's just hexagon and hexagons rings around the hexagon of interest. Does that make sense?

[00:07:56] JM: It does. I've seen presentations at conferences where Uber uses this hexagon, thing, hexagon like format to display information to the audience.

[00:08:10] KN: Yeah, geospatial data is really tricky to work with at scale. There was a whole era of Uber. I mentioned, we started with MySQL was our business database layer, technology of choice. We pivoted to Postgres. We did a live cutover, from MySQL to Postgres and our DevOps team did some phenomenal things of basically making it, so we can run this whole thing live. There was basically no downtime. We but we moved to Postgres, because of the post GIS, and geospatial functionality creating functionality that Postgres offered.

Ironically, it was funny, that lasted, I would say, four to six quarters somewhere in there. We actually ended up moving back, because we hit a point of a Postgres just wouldn't scale for us. Through a friend of a friend. I got connected to these guys who were consultants, who were Postgres core code base committers. They hit me the whole career of a fine tuning and performance tuning Postgres clusters.

They came into the session with us. We started to have them on retainer. They were basically extended members of the team. They were running out of ideas. We tried literally everything, including talking to the guys who wrote it to scale up Postgres and at some point, we just realized this wasn't the right technology for us, which is when we move back to a much more classic MySQL, Hadoop, and eventually that evolved to HBase. I'm actually not sure what they're using now, but a much more classic MySQL back stack, and then tried to solve the geospatial querying problem with a hexagon tessellation structure instead.

[00:09:55] JM: The choice of database and data infrastructure you just gave one example of how a conventional choice basically broke at scale.

[00:10:06] KN: Sure.

[00:10:07] JM: What were some other ways where off the shelf solutions were breaking, and you had to recreate your infrastructure strategy?

[00:10:15] KN: Yeah, I will say that Uber philosophically, and I think this might have come right from the top, rate from Travis some sort of approach. Tended to buy us to build over by. We bought very few technologies off the shelf. One of the most obvious and big ones, where we had this extra little dependency was on Google Maps. The Google contracts was renegotiated every 12 or 18 months, it seemed my data guys were tangentially involved, kind of supplying the quantitative figures to all of that.

Eventually, we realized that paying Google your order of 10's of cents, every time a Google Map popped up, was a strategic dependency we didn't want to take on. That began the - we don't want to buy this anymore. The classic build versus buy manifested itself in us going out and buying our own mapping product, and developing an in house mapping solution. That I think was initially - let's just figure out a way to attack marginal cost, but also created some really interesting opportunities to do embedded traffic mothering, vertical integration of things the operations team knows that there's a parade going on, or these roads are closed, and you could develop a tool such that the ops team can close a road.

It cascades all the way down to the stack and the reading engine becomes aware of it and the drivers know not to go on. That deep vertical integration, I think was a lot of what enabled our scalability and operational leverage without having engineers in the loop trying to like submit pull requests, or the, the map graph if you will.

[00:12:00] JM: The whole build over by decision. To my mind, this is a point of differentiation from Lyft, I think. Didn't like Uber went more heavily into colos, building their own infra, even having more of an open source presence, I think, like Lyft has definitely built up an open source presence of its own. I feel Uber went a little more aggressively into that direction.

You have been a much more active engineering blog. Much more active interface with the public engineering audience I felt from Uber. The differentiation it had implications for how the company developed differentiation cost structure, because Lyft is on AWS. Lyft I believe, struck some long range partnership with Google or with alphabet around mapping, probably feeding

data that helps Waymo, or something like that. Much different economic strategy. Do you have any perspective on how what are the costs and benefits of that strategic decision? Or collection decisions?

[00:13:09] KN: Yeah, of course. I think, so let me start by saying that I don't think it was always rational. I don't necessarily always agree with Ubers philosophy. I did think it made some kinds of sense in that. Travis, I would say from the beginning, was very much an entrepreneur that had a very clear understanding of what are the things that I am uniquely great at? What are the things that I find like uniquely aggravating? Or what is my point of view on the types of experiences we need to avoid? Then how do I do a really good job of propagating that into every decision the company makes?

One of the things I think Travis was an unqualified genius that was fundraising, and selling. What we ended up with was a mindset of, we could find top decile talent. We could find money on a consistent basis to both pay for top of market talent, but pay for a lot of it. When we came to these build versus buy attitudes, there's always whenever you're doing build versus buy, and on the buy side of the equation, it's always some fundamental trade off in my opinion of, we can substitute a bunch of development time and get a product which you bring something forward from the future to present day, that the upside to all of this.

The challenge is that the interfaces might be a little clunky. There's something that which is slightly imperfect or it doesn't quite scale the way we want to. There's always a constraint, which usually manifests itself in clunky flow or clunky scaling and I think Travis from the beginning was obsessed with this magic experience. These clunky interfaces drove him nuts. I had to build search screens, those little pop ups that would happen and he would be on me about loading times for this little like modal that everybody was just clicking through anyway.

He was like obsessive about that stuff. Whenever we were buying tech, as soon as there was a clunky interface, or something that was 95%, great. Travis will always say, “Look, like we have the money to just go find the engineers or go build something, which is your 98 or 99% great.” I think in some ways, the build versus buy decision kind of grew out of his obsessional product experience and a pragmatic viewpoint of, at the time, money was straightforward, recruiting top tier talent was straightforward. So, why not at least take a stab at building.

I think it definitely gave us the opportunity early on, to start to do some uniquely great things. If we were the first people to really build out some of these flows, these double binary flows, deeply integrated, mobile AV testing flows. I think that enabled us to innovate in ways, which other people struggle to replicate. The cons to it is that over time, all of these products require maintenance and support and continue to duration. You start to have this large ongoing support cost, manifested itself in both engineering salary dollars but also opportunity costs, what those engineers could be working on.

Also the motivation challenges of hiring a bunch of people promising them to keep building new and innovative stuff, and they're working on version 17 of the same thing, they've been working on. I think that all started to bog down a bit. I think, I haven't done with Uber since end of 2017. I think in the interim, they have unwound some of these build versus buy decisions and call back to buying. I think that was probably smart. I usually think there's a lot of wisdom and in build the things which are core to differentiating your product, but also recognize not everything needs to be built in.

There's a smart thing to be said about buying or even build, and then buy as the services or third party product catches up to your company but yeah, I think it's a nuanced decision. I agree with most of it, maybe 90% of it.

[00:17:37] JM: As far as an application of data science, the economics of the business has been thrilling to watch over time. Or just from a human incentive point of view, you start with this, $5 per ride, highly subsidized thing, and you have been able to boil the frog over time. Especially during and post pandemic, the economics of Uber look just so appealing compared to how they used to be in the very subsidized world. Now, it's like, I will effectively pay taxi prices or premium on taxi prices to just get the Uber service.

I know I'm not alone in that regard. Was there always a plan around this? In the data science around the economics of pricing. I'm not sure how heavily involved you were. If you were involved, can you help me understand the strategy for how to ratchet prices and how to maintain a healthy range of prices? Or how you approach that problem?

[00:18:45] KN: Sure. Yeah, absolutely. I was very involved with this. I think we discussed prior to this, I did the first couple years of dynamic pricing. I in a way invented surge, your welcome to the world. It's an interesting problem. Even from when I joined. I was referred to Uber from a friend. The tagline was like, Limo Company with an app with big dreams. It was black cars only at the time. I was like, “Okay. Interesting, I'll go check it out.” Even at 2011, there was this idea of, there's a transportation layer in every major city in America, that doesn't work particularly well.
We want to start with his limo service, but eventually have a point of view on all of it. I think that that larger vision is important. Let me obviously worked on business side, but it helps reframe the problem mathematically like it's not just about getting town cards to people efficiently. It's about fundamentally increasing the throughput of the transportation in an automobile layer in every city and public transit for that matter.

That starts with, okay, cars rideshare runs really efficiently. In the abstract, you realize it's a problem of how do we just move stuff in a cost and time efficient way. It's an optimization of cubic footage within an automobile, per time, per hour on dollar, cost optimized basis. I think dynamic pricing came into it relatively early, where what you figured out without pricing, we had this era where you couldn't get Ubers on Friday and Saturday nights.

We tried everything to onboard cars. The operations teams did these herculean driver on boarding’s and it onboard 50 drivers one week, and then 150, the next week and 500 the next week, and we were still maxing out. Travis started talking, we started drifting news like a standing Friday meeting about we should figure out a way to price Ubers fairly, such that there's always a number available. That was one of the things we figured out really early on was key to a magic Uber experience was, if you open the app, and there are no cars, you get this viscerally negative reaction.

Always have a car available. We tracked the, we called them zeros, the number of sessions where user didn't see a car. There were zero cars available, was one of our like, single biggest negative KPIs. That number had to be zero every week, or we were doing our job. Pricing became a big part of that. What was interesting about it was raising prices, we had this system that I developed that was hokey, and I’ll talk about in a sec, but raising prices did three things.

One, it disincentivized demand that there were people who said, “You know what? I'm willing to walk. I'm willing to wait a couple minutes. Let's go have another round, or we'll order dessert and wait for Uber prices to come down.” Secondly, it's incentivized supply in time and space. There was this idea of – we saw this, most obviously in New York. If you are a driver who's sitting in a suburban in Brooklyn and you know it’s Friday night and there's demand in midtown, or downtown Manhattan, you're still looking at this and saying, “God, I got to cross the Brooklyn Bridge. This is a massive pain in the neck for a very uncertain reward. I'm not even going to bother. I'll just wait for the next Brooklyn ride, kind of thing.”
Being able to raise prices spatially, geospatial dynamic pricing, was a way of playing with the calculus on the risk reward of doing something, like relocating across the city, incurring gas cost, incurring the headache of just fighting traffic to go downtown. It helped drivers move to places of higher demand, or more importantly, underserved demand in the city. It also helped them train their schedules to say, “Okay, I'm going to work later hours. I'm going to shift my schedule from more downtime to more busy time, just because the economics are better on that.”

The third and the most subtle thing about it, was it actually increased the throughput of our system. I think, what was fascinating about it is what happens when Uber's get too busy in the marketplace. I've done a couple talks on this. When you hit a tipping point where cars become less dense, they have a lower spatial density and ETAs go up. You've probably seen this in a really busy time. Your Uber is not three minutes away, it's 15 and 20 minutes away. If you request that car, your transaction takes longer in time. It takes the car now 15 minutes to get to you, plus, the 15 minutes from where you're going to where you want to get dropped off. Your transaction took 30 minutes.

Whereas, if you call during a slow time, when the car was one minute away of the one minute to get to you, plus 15 minutes to go to your destination, your transaction took 15, 16 minutes. There's a tricky second order effect that happens where in times of high demand with demand outstrip supply, transactions take longer, and the system is actually less capable of serving the transactions. You end up with this negative feedback loop, which causes Ubers to spike to a 100% utilization.

By keeping the system out of that negative feedback state, you actually ended up doing – It was interesting. When we’re in the study, and it was like, for every 0.25X multiple increase in pricing,

you had 25% higher per fair value, obviously. You also have 71% higher revenue per hour, because of the marginal increase in the throughput of the system, by staying out of that destructive feedback state system.

We're saying that pricing helps keep Ubers available, and that that was one of the big talking points back then. It was actually literally about maximizing the number of trips per hour the fleet conserve, because otherwise, you end up in this – the fleet can do roughly half as many trips between 90% and a 100% utilization as between 80% and 90% utilization.
We had this whole system. You asked me how we built it, what the focus was on. That was the single, biggest mathematical insight, is there's a tipping point in utilization, where we start to get negative feedback signals. You have this min-max problem, if you want to be as busy as possible without falling into a negative feedback loop. It's actually funny, the very first models were literally univariant, and it was a PID loop, where you would just say, tipping utilization, measuring utilization. If measuring utilization is within 10% of tipping utilization, raise prices, step up.

You would have a scorecard of how many steps with the energy chunks. First, it was 0.25X, then it was a tenth of a point 0.1X. Every five minutes, they would run this calculation, dial prices up or down. The operations teams could do these, basically set guardrails. We built this whole internal tool. Again, we're big on automated internal tools, where you could say, I want to vary prices dynamically from a floor to a ceiling. It was usually 1X to 2 or 3X. They can change a little bit of, I wanted to be really sensitive to supply shocks, or demand shocks. You have the heavy weight, the derivative term.

They had a couple. It was three or four options on the control panel. Click a button, then the system would take over and just vary prices up and down. It worked surprisingly well. I mean, for a model as simple as measuring utilization, feed it through an algebraic expression, that worked for years. Honestly, for the first 18 months, we're busy scaling this out to every city and doing all these different product lines.

Eventually, you get a much more sophisticated approach that people, I think, who have published the most research on this are actually DD, where the most of his approaches is you treat this as a – you basically calculate the expected value of every driver, rider pair, see a

bunch of requests come in. They sit there for 15, or 20 seconds. I think, it's usually 15. You have a bunch of drivers who become available. You compute the possible permutations of every driver pairing and find the configuration which maximizes net present value and net future value, both of which have certain model constraints.

[00:28:18] JM: Do you have the time to do that? The time it takes for the rider to – because you're describing stable matching, basically. Right?

[00:28:25] KN: Right, exactly. What's interesting is there's actually a floor. There's such a thing as too convenient in the rideshare space, where we discovered the probability of cancellation goes up from about 90 seconds to 30 seconds. You have this. We got a bunch of these right. You see, obviously, a probability of cancellation goes up as ETAs go up. You're like, “Oh, screw it. Uber’s 15 minutes away. Never mind.” You have the falling probability to about 90 seconds, and then it spikes again.

What we've hypothesized, we called, a bunch of people actually verified this in user research is it's like, “Oh, my God. The Uber’s here. I have to put my shoes on. I wasn't ready to go. I was expecting three to five minutes.” It has to be fast, or it has to be in fact, our real-time systems, our dispatching systems have performance constraints. I think they had to execute stable matching and think it was something like, 750 milliseconds or something like that. You do have a little bit of time. You can basically batch people for five or 10 seconds, run the stable matching algorithm in about a second. Actually, give them the car in 15 to 20 seconds.

Users don't seem to mind. In fact, they like it. There's a little bit of intentional lag. Sometimes, it's mobile phone lags. Sometimes it's just convenient before you get the car sent to you. I think, sometimes you can do stable matching, or straightforward just by integer number of requests, and you just say, “Okay, every 10 requests, we're going to dispatch.” We've experimented with that. I don't quite know the status of the dispatching algorithm is. Again, my intel is a little stale.

Early on, we honestly just ran it as univariant PID. Then you started to do a little bit of waiting function with geospatial adjacencies between these different neighborhoods we priced. Well, the big superpower to them is that they are autonomous. The big drawback to them is that they are autonomous. We consistently got tripped up famously early on, and I think we've since

recovered a bit of raising prices, or dynamically pricing in circumstances, where your price optimization was not the highest order bit.

The most famous examples were the Navy Yard shooting in Boston, or there was a café, there was a mass casualty event in Sydney, I remember at one point. The system was responding. The model itself was responding in a way, which makes economically rational sense, which is a bunch of people want to leave this area. The price of an Uber should go up, but it sidesteps common sense, human decency and ethics and your price optimization is not the highest order bit for the company at that point.
We always got tripped up with this fundamentally reactive nature of how our models were built. I think, you can shorten that feedback loop to turn it off in 30 seconds, not 30 minutes. I don't quite know how you get to proactive root cause analysis of what is causing the pricing driving, the price spiking, if that makes sense.

[00:31:50] JM: Okay, let's shift the conversation a bit, given that you're now an investor. We've been talking mostly at a pretty high-level, relative to the data infrastructure problems that the data engineers – burgeoning data engineering ecosystem was being tackled at Uber while you were there. You saw when you were at Uber, the rise of a lot of internal technologies that have now been productized, whether we're talking about M3, which is now Chronosphere, or the geospatial technology, which got productized, has unfolded, the data lake middleware that is at least getting turned into open source projects.

I'd love to know what are some of the broader product areas that you see opportunities for companies to be built in? As far as I'm concerned, after crypto data engineering is basically, the coolest space to be building in. It's the most problem rich. Maybe I'm exaggerating. I think, data engineering is a really great space to be investing in. I'd love to hear about the companies and the problems, the spaces, the subspaces within data engineering that you're most excited about.

[00:33:05] KN: Yeah, absolutely. One of the things I think we built at Uber, we're in the beginning stages of building it, but I haven't really seen productized that. I think, obviously, it's been nice for me as an investor with a bunch of the old Uber data gang starting companies, including the Michelangelo guys with Tech Tom and a couple other things.

One interesting part that we were trying to struggle with was how do you balance data security, and data access, that's the need for creativity innovation. The system we tried to build at Uber was a principle of basically, everybody could theoretically have access to every piece of information at the company. You have a spectrum of friction, which is ideally correlated well with the riskiness, or anomalous behavior, like how weird is this request, given everything we know about you and your job title and your job family.

I really like to see somebody explore that idea in a product around data security and data access. It's a little on the edge of data engineering, and I'll wear that. The idea of being, if you're a driver operations manager in Phoenix, and you're accessing client records in Boston, that's very weird. You have very little business reason for doing so. Somebody should be made aware of this, or you should have something where you say like, "Please explain why you're doing this. Submit a request for approval. Somebody has to authorize it."

We also had lots of very creative people who we go adjacent to their job title, or exploring something else. Sometimes, anomalous data access patterns were a great way for us to do internal tooling and product research, and just saying like, "Hey, you're pulling a bunch of different driver ticket information. Why are we doing this?" They're trying to create the ability to think through what would a insurance, or risk product look like if we wanted to build it internally. Some of our best ideas just came from people riffing on tools.

[00:35:24] JM: You're talking about like a porous, zero-trust detection service for self-serve data engineering.

[00:35:32] KN: Right. I think, the core piece of it is the ability to build a data access risk model that has correlations of basically, data access patterns, and how, given this pattern in the context of all of the rest of your patterns, how weird is this? If you could build that middleware, I think they're an interesting product.

[00:35:53] JM: All right. If somebody in the audience wants to build this, you have two co-investors right now. That's a cool idea. I would totally invest in that.

[00:36:03] KN: Yeah. That's certainly one. The other stuff I've seen in this space, and it still blows my mind that somebody hasn't solved those yet, is just a really useful query runner. We had this product inside Uber called Query Builder. It was literally just a text box, but you had templated queries. It was deeply integrated into a data book. It had unique data runs. What was so great about it is it basically, intermediated the SQL query box. If you were on a laptop, and you just fired up MySQL from the prompt, you can basically do whatever you wanted with the company's data.

You had that ability to do is for power users, but it was this product, which enabled basically, everybody at the company who knew a little bit of SQL. You give them a six-week SQL bootcamp that they could do asynchronously. With that and Query Builder, they had basically access to all of the company's data technologies. It blows my mind that there isn't a product like that, that is not standalone.

Looker does this a little bit, but it's within the Looker ecosystem. Metabase is probably the closest I've seen to doing this. I just really wish we could build a really simple standalone query runner. I don't know if it's a billion-dollar company, but it boggles my mind that literally nothing exists like this. It's not bolted into somebody else's bloatware product, in my opinion.

[00:37:41] JM: Okay. You've given some requests for startup thing. What about stuff you are actually seeing in the market, whether it's reverse ETL, or data lake middleware, or something like that? What are the sub-categories in data engineering that you think are exciting?

[00:38:03] KN: Yeah. I think, there's a mean to see a whole ecosystem of tooling built around the concepts of data democratization behind products like DBT. Because I think they’re pretty interesting, especially in that gap from I feel like right now, the state of data pipelining is, you have this infinite scale, data lake end of the spectrum. You have your databases at the business, or application layer. The pipelining is becoming increasingly straightforward.

Like with DBT is basically, if you don't SQL and a little bit of light scripting, coding, which describes a much larger population of data professionals than some of the prior vintages of technology. If you know that, you can basically build a data pipeline, and you're unblocked. As

soon as it gets to the application layer database, there's this massive context switch into your dashboarding tool of choice, or your last mile tool of choice.

I'm seeing a lot of companies. I think, DBT is very smart about this, that they're starting to build APIs, such that you can basically access your interface, or access your DAG rather, through a machine interface. A lot of DBT-adjacent tools, which I think is really powerful. There's one, I haven't done the investment yet, but just to tease it, where you can basically build dashboards totally customized using DBT-like language.

There's a company called Snap Data, which is doing something similar, where you could basically do conceptual business integrations of like, “I want Stripe consumer data pipeline all the way to Looker, through a DBT graph.” They give you a template. Those kinds of things, like start with DBT as a concept and then build, finish solutions, or adjacent products, I think are really compelling.

I think, the other thing going in a totally different direction, because my last big thing at Uber was working on Uber Freight, which was part of Uber ATG, which meant I hung out with a lot of the autonomous car guys, and a lot of people working on CV and visual inference; is this serverless, high-performance GPU compute. I think, GPUs are everyone's figured out like, “Oh, yes. This is a good idea in the ecosystem of products at the hardcore data infrastructure and data engineering level is coming along.”

I'm really eager to see what happens if you can basically, enable any data analyst, or any data scientist to safely and responsibly run any job they want on your GPU cluster without the data engineering glue to get from idea to infrastructure. Anything around your serverless GPU, I think is really compelling right now. Frankly, I just think it's still too hard and there are a bunch of underappreciated opportunities in the space. You see autonomous vehicles and trucks, absolutely. Retail coming along. You've got the standard cognitions in Amazon goes over the world.

I still think images and videos are an underappreciated. They are undervalued for the amount of data that they can contain. I think, part of that is just figuring out ways to extract that information

is basically a visual inference problem that is not really well supported by current infrastructure to win.

[00:41:53] JM: Well said. The solution there to basically, high-dimensional data formats that are refined over time with additional metadata, or produce downstream data. You talk about images, or videos. You have images. You have a bunch of images in some database, or in some blob storage, and you want to do all kinds of refinements on that you. Same goes for video. There just is not good, best practices for managing that data, especially as it makes its way through data pipelines. There's lots of ad hoc infrastructure that gets built around those kinds of workflows, right?

[00:42:33] KN: That's right. I think the whole stack is a little bit underdeveloped, to be honest. Some of it is blob storage, some of it is query layer, some of it is getting it last mile to what is the business problem you're solving. It boggles my mind that I can't just do select star from Netflix_videos, where video contains concept Halle Berry and get a SQL-like response. That inference and ability to search, and organize and access information, I don't think it really exists to that degree of simplicity.

Obviously, there are ways you could get that, I'm hinting at. At the light level of the data scientist who may or may not be trusted with deploy privileges, that level of technical sophistication. That is a pretty heavy lift right now. I think that there are massive industries. I've spent a bunch of time in LA working in entertainment and gaming, where because their data formats are inherently visual, it's inherently not unstructured data, are being left by the wayside on some of these best practices, and all of the opportunities unlocked in things like FinTech and enterprise, and e-commerce, because the information is fundamentally more structured. I think, getting better at unstructured data as a whole, and I think it starts at the infrastructure. My hypothesis is it starts at infrastructure and GPU democratization is a massive opportunity.

[00:44:08] JM: Cool. Well, we're about at the end of our time. You're obviously doing investing right now. I'd like to close on the question of if you had to build a infrastructure company, or I guess, any company, what company would you build today?

[00:44:27] KN: Oh, good question. On the fun side of things, I invest predominantly at formation preceding seed. I spend my whole day talking to founders about the company they're starting. Sometimes the idea doesn't work, so I actually spent a lot of time thinking about this. The company I would probably be starting right now would be tackling, or running rate of the problem of data sharing and data movement between companies, that I think lots of companies out there are aware that data is important. It has value. The analysis you build on it intrinsically have value.

It's always constrained as a, what can we do within these four walls? I think, data sharing outside of companies, and I say, sharing in the most vague sense. This can also be data selling, is a very risky proposition, right? You have a very capped upside. Sometimes it's big, but it is finite, of how much money can I get for this data? Or how much reputational gain can I get for it? A pretty significant downside. If you screw up data sharing, you share the wrong info and there's PII, or you share a dataset that gets replicated infinitely and is immediately commoditized. You're basically going to get fired and your company could be sued, and there's a pretty catastrophic downside problem.
I think lots of folks, and I say this both generally, but also personally, both in my time in Uber, as chief data officer at Tala's are the same mindset. I always looked at this and was like, there's probably something here, but I'm really not going to stake my career on the choice of sharing this traffic information with somebody. It just feels the risk reward doesn't make sense. If I were to start a company, it would be trying to figure out how to shift the risk reward around data sharing. I think that some of that is security. I think it starts with companies that have data and want to push it. If you can figure out a way to make them feel safe and confident in their ability to share information, good things are going to happen.

There's way too much information siloed inside of these companies, and a lot of economic value. I mean, I know for a fact, Uber has a better road network, a better understanding of what the best performing bus stops and best performing rail stops are, than any municipal rail, or bus authority we work with and you see on the planet. That's not critiquing them. I just think it's an unfair comparison, right? We have the data of literally, hundreds, if not thousands of cities at this point. Of course, we're going to be able to figure out better signal than somebody who's just looking at their own slice of the planet.

[00:47:29] JM: Have you looked at that company? I think it's called Vanta? It's a ex-Coinbase, ex-AWS guy. Tim Wagner? He was the head of serverless at AWS, then he went to Coinbase for a while. I think, he was director of engineering there. He's trying to do enterprise blockchain done right. Basically, data sharing done right. Is that the solution? Is that what we're looking for?

[00:47:52] KN: I think, there's a security component to it. There's a company called Dia Hooks, that I saw came out. They were the last YC batch. I thought it was really clever, because they were tackling this from, how do we standardize and simplify the webhook creation process? If you want to communicate information via web hooks, we're just basically going to solve this entire thing and your engineers – The pitch is like, "It's two lines of code. Then you can do web hooks."

I think, there's partially that, too, of just how do you decrease the amount of engineering friction? What's interesting, I found in data license and in data sharing, is there's a lot of stakeholders involved. The engineers who are actually punching a pipeline out, but you've also got the biz dev in business stakeholders, who are brokering the deal. You've got legal risk and security who are making sure that we're not giving away the crown jewels. Sometimes that's risk and audit, sometimes that is literally, obfuscating this information, or contemplating, doing something like a synthetic data transfer.

There's a lot of stakeholders in this, part of why I think it hasn't been easily solved. When you are a company, and you say, "All right, I really – four business divisions together to go do something, to push data onto a marketplace, where the return value on it is pretty unknown." Very rarely do you have a customer who's willing to quote a price and wait for you to get your life together to do all of that. I just think it's a bunch of new avenues that never get explored.

My hypothesis is that if you can simplify that process, who start a flywheel of, a couple more companies are more interested and that pushes a couple companies over the line to like, "Yeah, let's do this." Which drives more demand, which validates the marketplace hypothesis, which engages more people. There's something there of just, if you could figure out what is the right series of features, and what is the right series of scope, and I think you're right. Vanta is part of it, but not all of it. You can unlock something big. That's what I'd spend the next five years of my life on, if I wasn't building and investing in founders who are doing stuff like that.

[00:50:08] JM: All right. Well, it's been a real pleasure talking to you. Thanks for coming on the show and sharing your experiences.

[00:50:13] KN: Of course. Happy to share. Thanks for having me.

[END]