EPISODE 828

[INTRODUCTION]

[0:00:00.3] JM: Satellite images contain vast quantities of data. By analyzing the contents of satellite images over time, we can identify trends in weather, soil and agriculture. If we combine that data with ground level sensors, we can gather a clearer understanding of how chemicals in the air, or in the dirt map to how things look from above via satellite.

Descartes Labs is a company that gathers high-dimensional data about our planet and turns it into machine learning models to be used by customers. In order to do this, the company has built out a data pipeline involving queuing systems, machine learning frameworks and internal tools that are used to aggregate clean model and measure data. Tim Kelton is a Co-Founder of Descartes Labs and he joins the show to discuss the high volume of data and the distributed systems that make up the Descartes Labs infrastructure.

[SPONSOR MESSAGE]

[0:01:08.1] JM: For almost 20 years, techmeme.com has been a common site that people in Silicon Valley go to for news. You might have heard of Techmeme. You might go there multiple times a day to catch up on what you missed. Mark Zuckerberg has reportedly been somebody who reads Techmeme on a regular basis.

Did you know that Techmeme has a podcast? It's called the Techmeme Ride Home and it's similar to Techmeme as a written content source. Techmeme catches you up on the latest news in the world of tech and the Techmem Ride Home Podcast does that every day at 5 p.m. Eastern in short 15 to 20-minute episodes.

The Techmeme Ride Home podcast tells you what happened in tech while you head home after a long day. It makes a great compliment to Software Engineering Daily, because most of our shows are pretty deep in the weeds, not really timely. We are pretty late to cover a lot of different topics. If you're looking for a source of tech news that is up-to-date, search your podcast app for Techmeme Ride Home, or just a ride home and subscribe to the Techmeme Ride Home podcast.

[INTERVIEW]

[0:02:39.5] JM: Tim Kelton, welcome to Software Engineering Daily.

[0:02:41.2] TK: Thanks. It's great to be here. You work at Descartes Labs, which is I think of it as a data platform, a place where you have data being hoovered in and gathered and it's accessible to developers that want to build on top of that data. This data comes from sensors and satellite data and other places. Can you describe the path that a data point would take from a sensor sitting somewhere, maybe it's in a field, or wherever you want to describe from, to the end-user who's maybe a data scientist, or an application developer?

[0:03:14.1] TK: Yeah. It's an exciting journey that's changing quite a bit and in some ways, machine learning is a big part of why that's changing. We focus on what we call remote sensing. Sensors that are on satellites, or aircraft, or high-altitude balloons, those types of things, and they're measuring the earth and they're measuring changes that are happening on the earth. Most people think of that as traditional satellite imagery; you're the analyst, you sit down, you look at a beautiful picture, you maybe do something on the desktop. That's how that's been for the last 40 years.

What's changing is huge amounts of data never ever get touched by anyone. Machine learning gives us the ability to build models, test those models, back test them, historically, how did they do in different scenarios? That path, part of it is the same in how the sensors work. They're usually mounted to satellite. They're pushed out into space. Those sensors are increasingly measuring more and more different areas, not just red, green and blue wavelengths of light. We measure things like water and moisture content, measure things like thermal and heat, can measure things like global height, even things like radio frequencies. Those are being captured continuously as they're orbiting the earth.

Then as that sensor goes over, say a ground station or something like that, that ground station will downlink and it'll send the data down to those ground stations. Often, there's many, many ground stations all over the world. That's where Descartes Labs, where our workflow really starts at that time. We normalize all of that data. Whether it's taken from a NASA satellite that was launched 20 years ago and is still up there collecting data, those are pretty old sensors, but they still have a lot of value. Or whether something that was the European Space Agency just launched this week; we want to be able to use all of those in a normalized API and make that as easy as possible for our machine learning team to build models and fuse those different data sets together.

We do a lot of different processing workloads to get that data to that format. Things that we have to do are the top of the earth's atmosphere and how that impacts the surface reflectance of the earth, where the sun is in position to the camera and the camera's angles; we have to take all that into consideration. Then we have to get the exact geographic location and make sure all those pixels from all those different sensors over all those years of technology all line up exactly right. Then you still have pesky things like clouds and cloud cover that can be between the satellite and the earth and mess up the measurements.

That's all in the processing pipeline. Then we run the models. That's a little bit more of the machine learning modeling type frameworks. We use things like Kubeflow. There are Kubeflow pipelines, those types of tools. Then often, the output we're not giving our end-user 20 terabytes of imagery to download on their laptop and go through for the next five years. We're giving them a summary of what the model actually told them; things that changed in the earth. Those could be how much corn was grown in Iowa this week, or how does that look this year versus last year? What are the water levels across the whole earth on all the reservoirs? Or is there fire every five minutes. We have a model that runs and looks for forest fires starting and things like that. How many solar panels deployed in say, California earth.

That's often the very end-product of what we produce. Not a set of imagery for an end-user. It's a little different than what people think and it's changing very quickly. It's really because of machine learning and being able to build and test those models through those big, historical archives.

[0:07:21.9] JM: One reason I was excited to talk to you is if I was completely out of work right now, I had no idea where to start, I think something that is an emergent opportunity for engineers is the fact that these data platforms are emerging. We've had the cloud platforms, they've had 10 years to mature basically and they're so good at this point. The APIs are getting so high-level. It's becoming less of an issue how do you build something, and more of a question of what do you want to build?

Obviously, one of the biggest opportunities is all these machine learning applications. It's like a bottomless well of applications that could be built. I'm sitting on a chair right now. Machine learning was probably not used in the design of this chair; it should have been. That's how many opportunities there are. Unfortunately, most of the data that would be used to build these kinds of applications is siloed in companies that already have enough opportunity, like whoever made this chair, whether it's IKEA or whatever. I'm sure IKEA is working on machine learning stuff, but how far along are they? Probably not that far. I mean, or they're far in one specific domain and it would be great if all this data was available for other people to use. Why has it taken so long for these data platforms to emerge?

[0:08:44.1] TK: To some extent, it was the individual end-user. In your example of IKEA, those are the types of use cases we're starting to build models. You can think IKEA is making a chair, but there's all the raw materials that went into the chair way before it ever – there is wood, or metal, or steel. Those are the types of models that were just really, really hard to do on a laptop.

You couldn't build and bring that much data right there. There wasn't that many CPU cores. When I was at Los Alamos, we had access to all these big high-performance supercomputing centers with thousands and thousands of cores, but you also need all of the data right there. You often need test data sets that you can train and test and label for machine learning in those types of use cases. Those weren't how systems were built.

We've taken a view that we designed it, our data platform, just solely for machine learning workflows and optimizing for machine learning modeling time and being able to iterate as quickly as possible in models across a range of domains. Not just one specific use case, or one domain. We feel remote sensing of the earth and those types of datasets, there's more use cases than we'll be able to make products for. More end-developers are going to have ideas that we never thought of. They'll want some of the work we've done on these ingest processing pipelines to get that data all cleaned up and normalized.

We can make them incredibly efficient and make them able to build those types of models and things we just don't have the bandwidth, or haven't thought of ourselves, or don't have customers. I really see a machine learning as the driver and being able to build and test those models really, really quickly; maybe one example that we're talking about is when we built our first agricultural crop models that were across the whole US, they pulled a basically imagery from every farm, every field, everywhere in the US every day. It took over a 1,000 model iterations to get a really accurate version that was good.

Then you need to test that backwards through time. That was tested over 15 years of historical imagery. It's quadrillions of pixels of imagery that have to go through those models. You need the cloud compute, you need the storage, you need the network, you need all of those things to scale right there.

[SPONSOR MESSAGE]

[0:11:30.8] JM: Heroku is a large-scale cloud infrastructure provider that has many interesting challenges, including the challenge of how to run relational database workloads, such as PostgreSQL.

In tomorrow's episode, John Daniel from Heroku joins the show to describe the challenges and the opportunities of running database workloads in the cloud. He discusses the Heroku deployment strategy, the DevOps strategy, the infrastructure engineering and the culture of Heroku’s engineering environment.

It was a great show with Heroku. I'm proud to have Heroku as a sponsor of Software Engineering Daily. I hope you enjoyed it as well. Tune in tomorrow. You can also listen to our past episodes with Heroku engineers. We've done shows on Heroku Kafka, Heroku continuous integration and several other Heroku engineering problems and products. These are great shows with very talented engineers. Thanks again to Heroku for being a sponsor.

[INTERVIEW CONTINUED]

[0:12:45.3] JM: We've had SafeGraph on the show. They're another data company. They are also starting with location, geometric data. Why is that a good place to start as a data marketplace? Why is the area of mapping and location data such a good place to start, of all – Why not go into chair data, for example?

[0:13:10.4] TK: Well, I don't know a whole lot about chair data. I do know for my personal use, I use – I work out a lot. I have a Garmin on my mountain bike. It's measuring all these different values; it's measuring my heart rate, it's measuring my blood and my lactate thresholds, it's measuring the altitude, it's measuring all these different sensors, but it's all changing as I'm moving around.

I'm just putting on myself more and more sensors. I think more and more IoT type sensors and more and more just activity-based sensors. Just what we care about as humans is relevant to where we are on the earth. I don't necessarily care about traffic in New York right now, because we're in San Francisco. I do care a lot about traffic, or maybe the lines at next to get in the door, or things like that are very, very relevant to me right now. I think that's going to only increase as sensors become easier to deploy and are just easier for us to have with us all the time.

[0:14:17.4] JM: Then why not start with health data?

[0:14:20.1] TK: Well, our background, like I myself and the founding team of Descartes were scientists and researchers at Los Alamos National Laboratory. We have worked in this area of remote sensing and large-scale HPC. For us, the answer is easy. We're not health physicists. Instead, we have backgrounds in this area.

[0:14:42.7] JM: Interesting. Yeah. I guess, your first answer speaks to the domain expertise that’s required. If you have to normalize for clouds and the physics of how clouds affect light, or something, that's a pretty big moat for in terms of a data platform builder.

[0:15:00.7] TK: Yeah. There's not a lot of fields that parallel, or that your experience really works to cloud cover and those types of things, but one area we have a lot of our team is astrophysicists and cosmologists. That's the same problem. They're just looking up at the sky with telescopes, instead of looking down, but they have all of the same challenges.

Again, that's also where they're measuring pixels and that's what we do as well often. It's not building a data platform that's really good at analyzing logs from your Apache server. That's what Hadoop and Spark and those tools were really built for and excel at. Whereas, the geospatial world is very instrument specific often. NASA, some of these satellites, they would work on for years and years and then these things would cost hundreds of millions of dollars. Now they're getting more and more commoditized and they're getting to be these small, cheaper CubeSats. A lot of the codebase is very, very scientifically based. Lots of C and Fortran and most of our stuff is all Python, but that whole scientific codebase that goes with the sensor and how the sensor was exactly calibrated.

Also, just space has a lot of different challenges that we don't have here on earth, in terms of boards being fried, or you can't just do diagnostics or switch something out. There's a lot of scientific code that gets paired with each sensor. It's not a trivial thing to just pick up. That's our goal is to make it so somebody that comes in with an understanding of say, in Tensorflow or scikit-learn, or something like that, and wants to make a model. We extend the numpy array, and so you just pass in a geo-JSON of this is the location and here's my start time and this is the end time and these are the wavelengths of light; maybe I just care about infrared, and return it back in a big array that I can start building and testing my model on.

It doesn't necessarily matter, like you don't have to know that NASA had this weird calibration problem with this one sensor on these years of imagery. They threw a firmware upgrade in 2006 and everything from then on is good. Before that, it's bad. It's incredibly hard for someone to just know that type of information.

[0:17:33.3] JM: As a developer, I encounter websites that are offering me tools of different types. In terms of how I can actually acquire those tools, I think there's a spectrum; on one end of the spectrum, they've got an open API, it's quite easy to integrate with, it's quite seamless, it doesn't require me filling out a contact form. On another end of the spectrum, you have these websites where it's a boat of marketing materials and a contact us button.

There's plenty of space in the gradient between those two extremes. As a data platform, I assume you have to fall somewhere in between that extreme; can't be a free API. I remember using the Yelp API in college and was free, at least up to some rate limiting area that didn't affect me. Say what you will about Yelp’s business model; their API for free consumption of location data by developers was completely open and unmercenarial. What's the ideal way to vend this stuff to developers?

[0:18:39.8] TK: I can say our approach is a lot of our costs get around egressing of data and we want to make it as easy as possible to run the models and come up with derivative type products native in the cloud. We're not building a platform for people to just download data per se, more to build models and build insights on top of that and efficiently as possible. We first just built that for a large percentage of our team at our company is what we call applied science. Most of them have PhDs and backgrounds in machine learning and remote sensing. We built all these APIs just for them. There's more use cases than what we have time, or bandwidth to do, so we've just started externalizing that.

We have tens of petabytes of data up there. There's a lot of cost of cleaning that and calibrating that and making that. Most of the value is running those models for customers. That's what we do as a company, if they don't know how to – if a company comes to us and say, “We're not real good with remote sensing. We know machine learning, but can you help us?” That's where we have teams of applied science – scientists that really do know remote sensing really well and can go really deep in that and build something fairly custom for a customer.

Your other comment earlier, a lot of customers have their own data archives. A lot of times, that's really where the real value is; combining that customer’s one unique piece of data that's specific to that vertical and then these big archives of remote sensing data and building those types of models that really are meaningful to them, but might not have meaning, or as much meaning to other people in other industries.

[0:20:28.9] JM: What I'm imagining the infrastructure that you would use to build this data platform, I can imagine data coming in, getting queued up in something like Kafka, or Google Cloud pub/sub. I imagine there's some stream processing systems that are reading that data off of a queue and then writing it back to the queue after being cleaned. Perhaps that happens again and again as you do these normalizations and iterations there. I can imagine, once it's cleaned in a certain state, you kick it to a machine learning system. Then eventually, maybe that is doing some more cleaning and then eventually you get to this point where okay, this data is refined enough. Can you tell me a little bit more about the workflow that goes on and some of the infrastructure tools you're using?

[0:21:14.6] TK: Yeah. You're right on the money. We use a messaging queue. We are heavy users of pub/sub as far as – but satellite imagery is –

[0:21:22.2] JM: Wait, wait. Let's talk about that real quick. Why did you go with Google Cloud pub/sub, instead of Kafka or managed Kafka, like Confluent?

[0:21:30.0] TK: Originally, our first queueing system that we used was a Python tool called Celery Python library and we use that with a Redis back-end store. That was very, very early days of GCP when we were just starting as a company. That was 2014. A lot of GCP that exists today didn't exist back then. I think pub/sub was very, very early in 2014. I had used Celery as a message queueing system.

Satellite imagery is not a pure streaming per se, where it's a very much more of an embarrassingly parallel type problem, where a lot of it's very bursty. A satellite goes over a ground station, sends down a chunk of new scenes that it's collected in the last few orbits, then you have those scenes come on and that would start triggering our processing pipeline. Those processing pipelines, the code bases can be all over the board.

European Space Agency does a lot of stuff with Java. We might have to have some Java library that's been all tested and has lots of scientific papers around it and we know it runs. Another piece of our pipeline might be C code. Or if we write it ourselves, it's usually going to be Python, so that's all it gets containerized. Then we use a task queuing system. Then of course, things like Kubernetes gives us more efficiency and tighter bin packing for those workloads. Some are real big memory hogs and others are real CPU-intensive. How do we optimize there and scale those out? Then we use cloud object storage and we serve all our data is just live on object storage. We don't use network attached storage or anything like that.

[0:23:21.6] JM: You use CDN or something?

[0:23:22.9] TK: No. Well, we have used a CDN, but you make a model for deforestation in Borneo and I make a model for Florida for hurricanes, the cache hit ratio isn't quite the same as –

[0:23:36.2] JM: Okay. All right, fair enough.

[0:23:37.1] TK: - a pop star goes live with a new photo and that gets cached all day long. If you're training a model over and over again, you might get some benefit on the cache.

[0:23:45.9] JM: By the way, I think it's interesting that the consumption is like a file that's just sitting in an object store, rather than hitting an API.

[0:23:55.8] TK: Ours is all APIs and what we –

[0:23:58.6] JM: Okay. I'm sorry. The end-user is not requesting the thing from storage, downloading a big CSV or some file, Tensorflow file.

[0:24:07.7] TK: In fact, we built our own file system, a high-speed, read-only file system for how we compress and store all the imagery. For example, some satellites might collect 10 or 15 different wavelengths of light. When you think of RGB, red, green and blue, but you might only care about the infrared parts of that sensor's scene. Instead of pulling down a 4 gig file and untiring it and unpacking it and all that, instead what our API lets you do is just call just the infrared bands over just this fraction of the object and just pull that back and return that back through the APIs.

We spend a lot of time on that early on. Because we've built our own metadata store and we processed all that imagery through, we know exactly which spectrums that's captured exactly, which the spatial boundaries of the scenes and exactly where all the data is. Then we can pull just the bits that are being requested back and expand them.

[SPONSOR MESSAGE]

[0:25:22.0] JM: We like to explore cutting-edge technology and the real-world solutions enabled by those technologies on Software Engineering Daily. Amazon re:MARS is a chance to hear directly from the leaders and innovators, who are using artificial intelligence to shape the future. Amazon re:MARS is a new global AI event on machine learning, automation robotics and space.

re:MARS is inspired by the exclusive MARS Conference. It will combine the latest forward-looking science with practical applications. You'll hear from thought leaders across science, academia and business. Speakers include Amazon Founder and CEO, Jeff Bezos; Landing AI Founder and CEO, Andrew Ng and more. You'll learn best practices that you can implement immediately through more than 100 sessions.

See demos of automation technologies, engage in hands-on labs and interact with real robots at the robotics showcase. When you leave, you will be up to speed with the latest AI trends and best practices. You'll also have the technical skills to apply AI to your own work. I am only comfortable using that term AI, because I do think this is a great place to apply machine learning to higher-level applications and build applications in these different domains. I think AI is a fitting abstraction to describe this conference as being focused on.

Amazon re:MARS will take place June 4th through 7th in Las Vegas. You can register today and get $400 off the ticket price. Go to remars.amazon.com and register using the promo code SEDAILY. I am really tempted to go to this conference. It sounds really cool. I am not sure if I'm going to be able to swing it, because I've got a bunch of conferences to go to this summer already, but I will definitely be thinking about it a lot.

That URL and promo code again is remars.amazon.com and you register using the promo code SEDAILY. Thanks to Amazon re:MARS for being a sponsor of Software Engineering Daily. This conference sounds pretty sweet.

[INTERVIEW CONTINUED]

[0:27:55.1] JM: Okay, sorry. I totally took you for a loop there. You were talking about the choice of Google Cloud pub/sub, and then I think it's worth exploring – Okay. See, I'm doing these shows about streaming and there's 50 streaming frameworks. Now there's five or six different queueing options to choose from. There's Apache Pulsar and Apache Kafka and Google Cloud pub/sub and etc., etc. Kinesis. I'm like, “I can't keep all these things in my head.”

Then the streaming frameworks, there’s Spark and Storm and Flink and Heron and data flow, and it's like, Apache Beam. It's like, “Can I use Apache Beam? I know it's an API spec. It's not actually a streaming framework.”

[0:28:39.6] TK: We built our own to some –

[0:28:40.8] JM: You built your own. Okay.

[0:28:42.6] TK: To some extent. I mean, there's not a pure magic solution for geospatial data.

[0:28:48.3] JM: Why not? Okay.

[0:28:51.7] TK: It gets back to all those sensor types are just so varied and different. For example, one satellite we pull in is NASA's MODIS. That was launched in 2000. Think of what your software stack looked in 2000. Then we have other satellites that are going live right now and will be launched in six months, or things like that. Our ingest pipelines need to be able to clean up and calibrate all of that breath.

Containers worked really well there. We know the amount of scenes coming in from the ground station and the scenes that need to go through these processing pipelines, each satellite might be different sets of different processing pipelines. To some extent, they follow the same pattern of here's the chunk of work that needs to get done, here's the destination where that needs to go, here's how we update all of our metadata stores to make it really quick and easy to query it, and then here's the end API, that no matter what of those sensors you as a modeler are trying to build on top of, you have a uniform API to call that.

That's what we wanted to do for ourselves to try and build and iterate models. It was just so much time to constantly – each person have to do these types of workflow. We built all these APIs for ourselves first, because we couldn't really find anything that worked really well. Then now we've exposed them externally, so other people can use them.

[0:30:23.9] JM: In terms of distributed streaming frameworks, are you using any of these things? Flink, Spark, Storm?

[0:30:28.7] TK: No.

[0:30:29.3] JM: Nothing?

[0:30:31.1] TK: No. We use pub/sub quite heavily and then everything's built around those types of things.

[0:30:35.3] JM: You just have Google – everything into Google Cloud pub/sub and then Python, just reading off of that?

[0:30:41.1] TK: Yeah. Or you just have listeners waiting for an event to happen and then we'll spin up this type of worker to do these types of processing.

[0:30:50.3] JM: I mean, is anything appealing to you about any of these things, any of these streaming frameworks, or it's all just, “Nah. We don’t really need it.”

[0:30:58.4] TK: There's bits and pieces of that.

[0:30:59.5] JM: I mean, Kubeflow, does Kubeflow use – Oh, you use Tensorflow. Okay.

[0:31:04.1] TK: What we're using is Kubeflow pipelines. We have tried –

[0:31:10.7] JM: This is like a CI/CD thing for machine learning models?

[0:31:14.7] TK: Yeah, a little bit. It's a directed acyclic graph processing engine.

[0:31:19.4] JM: Oh, okay. It’s like air flow thing? Workflow manager.

[0:31:24.3] TK: We've tried most of those. We've done air flow and, what was the one Spotify did a while? Luigi?

[0:31:30.5] JM: Luigi. Yeah.

[0:31:32.1] TK: Of course, we have lots of bash and shell scripts that we've had as very first generation type tools. We're doing Kubeflow. We've been working with Google quite a bit on Kubeflow pipelines for quite a while now.

[0:31:44.3] JM: Awesome.

[0:31:46.6] TK: The modeling pipelines, people often it's not just the classification. There's a customer might want you to put their information about maybe they've hand-labeled a training set that they want you to train off of. Then maybe one step in your processing pipeline you need to paralyze out and scale that processing. That pipeline is a way to capture all the software dependencies for every single stage. Then under the hood, that's using Argo to deploy Kubernetes containers. Every single stage in that directed acyclic

[0:32:19.6] JM: Wait, what is Argo?

[0:32:21.5] TK: Argo is a little bit of a dag-type processing on top of Kubernetes. Instead of having a language and declaring everything with a language, you can actually define some of the processing steps via actual containers. That works great if you have this one step that's it set a Java code and then you have some other things that are C code and then some other steps that are Python code.

[0:32:46.5] JM: Wow.

[0:32:47.7] TK: Being able to assemble all of those together and then version them and then how do you do discovery on that? How does someone new come in to your team and how do they find the pipeline that produced this result last year, or maybe you've been running it in production for two, three, four years. Some of our models now we've ran in production for many years. That's very hard.

[0:33:11.6] JM: Wait. Why is it any different than regular old service discovery?

[0:33:15.0] TK: In some ways, it's maybe not, but you think of that whole entire machine learning pipeline needs to basically get versions. Say, if I'm a new person who comes in and I said, “Oh, we did this experiment last year on trees tree forest fires and detecting them. It had these 12 processing steps. Each of them had code written around them. Then each of those processing steps had their own set of dependencies and their own version management.” How do this training set equals the results in the model? How do I make it as easy as possible to produce that?

In some ways, that's what Docker is doing with containers, right? They’re unioning together all these file system layers together and you say this version in this tag. That rolls up and gives you this container runtime with all those correctly union together steps. That's what this is from a processing pipeline perspective.

How do you store those types of things? How does someone come in? How do they find that model? Maybe they just need to update the error handling, or make a minor bug fix. How do they get that in? How do they run the test dataset and get the test result and make sure they can run the entire processing pipeline through end to end? Then containers may help you with the deploy side, so you don't have the software dependencies that you would have with the VM. We've worked quite a bit on those types of – as our machine learning team has grown, that becomes more and more of a challenge. How do you find the model that we're running in production here? How do we find the new experiments we're working on? Those types of problems.

[0:35:00.0] JM: Does Kubeflow address data lineage?

[0:35:03.2] TK: No.

[0:35:03.9] JM: It does not? Okay.

[0:35:05.0] TK: No. That part is outside. You can do an object storage. You can actually do things like versioning of the objects. Then you can reference that version of that object.

[0:35:16.1] JM: You would want to map?

[0:35:17.7] TK: In our case, we just reference our API.

[0:35:20.5] JM: Okay. In the lineage problem, I see is you should be able – I mean, if you really want to go deep, you should be able to have a data set. You should be able to know the entire data set and the model that has trained on that data set and how that model has evolved over time as the data set has evolved over time. I've not seen that really solved yet.

[0:35:40.8] TK: Yeah. That's a very hard problem and especially on-

[0:35:42.6] JM: Luckily with satellite data and public sensor data for crops and stuff, it doesn't seem like – That this is actually one advantage I see is the geographical satellite image. Somebody takes a picture from a satellite. To somebody, that's implicitly public information.

[0:35:59.7] TK: Right. Yeah. I mean, NASA makes most of this data publicly available. It's just sitting behind an FTP server somewhere. You have a problem that you're faced with. Like, say a forest fire happens somewhere and you then say, “Well, I need the temporal stack of this geographic area going back. I want to model out the change in fire fuel, or tree density over the last 20 years.” Now you got to start pulling all this data down, cleaning it all up. Then you start finally at the very end, you can start building and testing your model, which just probably was the idea you had at the beginning and that was what you're really trying to do.

[0:36:43.8] JM: We're almost out of time. I know you're a distributed systems guy. What's the most acute engineering problem you're working on right now?

[0:36:50.4] TK: Really, we're spending a lot of time on scaling up the reproducibility of our machine learning models. That's what we started the last six months with the Kubeflow pipelines and those types of things. That gets to be a really hard problem to do at scale. One of the models we were looking at running at would take a few thousand cores and we'd be running that for 33 days. How do you test those types of models and how do you make them reproducible and then how do you get – when you're running at that type of scale, it's quite expensive, so you're always looking for cost optimizations and I spend a lot of time on those types of problems.

[0:37:33.1] JM: You can down to the expo hall. I think there's probably eight cloud cost optimization dedicated companies. Talk about an industry like we never would have anticipated –

[0:37:44.2] TK: Billing.

[0:37:43.8] JM: - 20 years ago. Yeah, billing. It's basically like, yeah, we do machine learning on your cloud bills and there's eight of us. We're all platinum sponsors.

[0:37:52.9] TK: That didn't exist even two years ago, right? That but that was a really hard problem. I used to get all these CSV files that were pages and pages and pages long and throw them into BigQuery.

[0:38:04.1] JM: Oh, my God.

[0:38:06.5] TK: You laugh, but I actually did that for our cloud bill. It's like, "Oh, that's worth maybe a couple hundred bucks a month," or some fee for me never to have to deal with that, so I can focus on actually the hard problems I'm excited about, that we're really trying to solve, that we feel that we have an opportunity on. It's not always billing.

[0:38:27.5] JM: Hilarious. Okay, well I could talk to you for a lot longer. I got to do another interview now, but really good talking to you, Tim.

[0:38:32.3] TK: It is great to talk to you.

[0:38:33.6] JM: Okay, thanks.

[END]