EPISODE 629

[INTRODUCTION]

[0:00:00.3] JM: Machine learning models allow our applications to perform highly accurate inferences. A model can be used to classify a picture of a cat, or to predict what movie I might want to watch. Before a machine learning model can be used to make these inferences, the model must first be trained and deployed. In the training process, a machine learning model consumes a data set and learns from it. The training process can consume significant resources.

After that training process is over, you have a trained model that you need to get into production, and this is known as the deployment step. Deployment can be a hard problem. You're taking a program from a training environment to a production environment, and a lot can change between these two environments. In training, you are running your model, you're training your model in an environment where you have access to lots of resources, and it's a different machine than what it's running on in production. This can lead to compatibility issues.

If your model is trained in one environment and you deploy it to another, who's to say that it's going to run in that new environment? If your model serves a high volume of requests once it's in production, then you might need to add scalability to that model. In production, you also need all kinds of things like caching and monitoring and logging. Large companies like Netflix and Uber and Facebook have built their own internal systems to control the pipeline of getting a model from training into production.

Companies who are newer to machine learning can struggle with this deployment process. These companies usually don't have the resources to build their own machine learning platform like a Netflix or an Uber.

Diego Oppenheimer is the CEO of Algorithmia, which is a company that has built a system for automating machine learning deployments. This is the second cool product that Algorithmia has built, the first being the algorithms marketplace that we covered in an episode a few years ago. They still run that product and it's quite a fascinating product. I think if I wasn't running Software

© 2018 Software Engineering Daily

Engineering Daily today, I would certainly look at Algorithmia as a place to potentially build on top of, because it's this marketplace of different algorithms that you can access via API. This show is not about that algorithm as marketplace. It's about the deployment system that they've built, which leverages their previous system. In today's show, Diego describes the challenges of deploying machine learning models into production and how that deployment system was a natural complement to the algorithm's marketplace. Full disclosure, Algorithmia is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:02:57.1] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT Iternational is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization

platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.
DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW]

[0:05:20.6] JM: Diego Oppenheimer, welcome back to Software Engineering Daily.

[0:05:23.5] DO: Thank you very much for having me.

[0:05:25.3] JM: You are the CEO of Algorithmia. The last time we spoke, we talked about your algorithm marketplace. Some people use Algorithmicafor deploying their machine learning models and other people call those machine learning models. How has that marketplace evolved since we last spoke?

[0:05:44.9] DO: Absolutely. Interesting enough when you're building a marketplace, a lot of times people are going to talk about this chicken and egg problem, and how do you get supply, if you don't have demand, how do you get demand, if you don't have supply? One of the things that we have looked at is we said, “Well, we think there's a lot of supply out there and this is something that we covered last time. There's a lot of people who created algorithms, functions, models that they want to do.”

We spend a lot of time thinking about, “Well, how do we get the path to acquisitions as easy and as smoothly as possible from a software development perspective?” How does a developer essentially, could we get it down to git push, right? That's the meta way of approaching this. Essentially, we built a lot of tooling, a lot of UX, a lot of functionality so to make it as close to all you really need to do is from your terminal do a git push and then you can essentially publish out your algorithms functions and models.

Surprisingly, in the world of data science and machine learning this is a tool that didn't exist and really attracted. They'd assigned to some machine learning engineers, because of the difficulty

that they were having grabbing their Tensorflow models, or caret models, or scikit-learn models and actually being able to publish them and host them and run them in production. What the tooling that we initially built for easy acquisition, or for making the life of developer easy ended up being the actual gateway for people really looking at us and saying, “Hey, this is exactly how I want to do deployment and hosting because it makes the whole process so easy.”

[0:07:23.9] JM: You were seeing people use the algorithmic deployment system that you made for a marketplace. You were seeing people use it for just their own model, so deploying their own models and then consuming their own models that they deployed?

[0:07:39.5] DO: Yeah, exactly. This interesting concept is we saw this category of what we would call private algorithms start really rising and especially when it came to how much they were being used. We would look at the Algorithmia marketplace and we say, okay, so there's these algorithms that are being published and other people are consuming them, or models that are being consumed, but then there's this really large rising category of models that were being put into the platform, never published publicly to essentially the public index, but had a massive amount of consumption.

This idea of like, they were essentially using our tool as a private hosting service for their models and the volume was actually really impressive. We started exploring what that experience looked like and who those developers were and what they were doing and what we found was that there was this non-trivial amount of the scientists and machine learning engineers who were really struggling going from, “Hey, this works on my laptop,” to , “I can run this as a large scalable service that is needed for my applications, and look, happens to be that if I just put these things in private mode in Algorithmia, this is exactly what I needed.” It was one of this we had a general idea that this was going to happen, but actually at the speed and the quantity and volume at which it picked up was really the surprise.

[0:09:03.3] JM: The developers who were doing this, who were deploying their own models and then consuming their own models, what were the problems they were encountering with the other ways that they were trying to deploy their models into production?

[0:09:17.2] DO: Sure. There's a couple of different things that were happening. One, very different skill set. If you think about the skill set that a data scientist has, or the machine learning engineer has, highly trained and statistics, understanding what the new frameworks were, how to do data processing, but scale, APIs, monitoring, uptime, these are just things that are not really part of the lingo, or things that they've generally encountered.
Given the teams and how they're being built at these companies, they were essentially being put in charge of doing not only the data collection, structuring, modeling, ETL the whole thing, but then on top of that now they're expected to run services. This is just not something that they have experience for. They would be bringing in platform developers, or other engineers to actually build that out for them, which ends up with bespoke solutions, so you get these one-off serving platforms for specific models. Then you get the problem of when you're essentially throwing code over a wall for somebody else to productionize as you well – now, you certainly get bifurcation of code, updates that they'll make sense. It's just the software development process actually breaks down quite a bit.

What this does is by automating the production side of things, it allows the data scientist and machine learning engineers to continue having control over the full lifecycle without having to worry about how the production part really have to build it themselves. On top of that, we still give the monitoring tools and the scale and the security and things that the platform team would want from a back-end perspective.

[0:10:53.9] JM: As the CEO of the company, this puts you in an interesting position, because you were seeing that people were using the platform for one case that it was not originally designed for, but I'm sure there were plenty of people who were continuing to use it for the algorithm marketplace purpose. Was there any tension between thinking about those two use cases and trying to build a solution for both of those products at the same time?

[0:11:21.3] DO: The interesting part is that is actually the exact same mechanism and it's the exact same platform, right? This was just more a difference in behavior that we expected. We still need to acquire these models, we still need to have more of them. One of the things that we notice is that actually people will come to host their private models, but then also explore the marketplace to attach it as if it were to build out AI pipelines.

Really what would change to a certain degree was how the acquisition was going to happen. If you think about how people come to us we initially were like, “Okay, well they might come to us for a specific sentiment analysis algorithm, or for a clawed augmentation algorithm, or for some image processing algorithm,” but the truth was that we have a large amount of people that come to us to host their own models and then go out and explore the marketplace as our add-on. Which if you think about it, is really in-line with how platforms work, right? If you think about how they're most popular platforms out there that we work with, they start out with a platform and then they add add-on marketplaces, or add-on perspectives and this was not something by design, but natural behavior that ended up with our developer population, which is they came because we solved their immediate problem, which is how do I host this and then they found things to complement their workflows right there.

[0:12:51.1] JM: I won't talk about the engineering of this machine learning deployment system. As you said, whether people are deploying their models to be consumed by random people, or to be consumed by they themselves, it's very similar technology. If you think about a giant company like Amazon, when they started building their internal services, or they started doing their service-oriented architecture, the legendary memo to everybody in the company about how their services are going to be deployed, it did sound very much like this. You deploy your services, even if you as a team within a giant company like Amazon are going to be the only consumers of that platform.

You deployed in a way such that it could be externalized, so to the other people in the company could easily consume it. It sounds like that even, that principle extends to you building this marketplace of machine learning model deployment. I want to talk about that. If we're talking about machine learning model deployment, instead of just these normal restful services; people think about just deploying their own microservices, it seems like there are some stark differences between deploying those two types of services, and I think this is why people like Uber or Facebook, they create their own AI model pipeline, which I think consists of all this tooling around the deployment process, maybe even the training process. These AI pipelines at places like Uber or Facebook, what does that consist of?

[0:14:22.9] DO: Yeah, absolutely. I think it's worth mentioning that the world of there's a general desire of always having an all-in platform. The fact is is that training and productionizing, or

deploying, or operationalizing whatever the word you want to use there is very, very different. Core, core differences. For example, the training process, it's stateful, right? You need to keep state during the training process. It's usually single user, it's usually very long-running, from a per job perspective, it's very long-running, uses tons of GPUs and CPUs and gets distributed amongst many machines, uses a ton of data, right? Data locality and the amount of data used in that processes is really big.

On the flipside of that, if you start thinking about what the operationalization is, or what some people call inference of a model, which is the actual classification or prediction side of things, it's usually stateless versus stateful, it's short but very intense bursts of compute, and it's by definition usually a multi-user problem, right?

If you think about I have a fraud model that's running every credit card transaction at a time, right? There's millions upon millions of endpoints potentially hitting that model at the same time, or replicas of that model to process that transaction and get back a result, which lasts only very, very few seconds. If you boil down to the problem to its your bare-bare characteristics, the training and the operationalization side of things are very, very different. Designing a system that encapsulates both is going to be difficult, hard, or you're going to have to give up on one side or the other, right?

You can see that in the industry that we've seen, it's like there's been a specialization towards one side or the other. When talking about AI pipelines, or things like you mentioned, like the projects like Michelangelo, or Twitter's Deep Bird, or TRX at Google, or FBLearner Flow at Facebook, the project itself tends to be the encapsulation of both training systems and production systems, that those systems are actually quite separate. The main driver to get those things out is this idea that like, "Hey, we productionize one model, we productionize the second model, we production a third one." Like, "Oh, we might be doing this a lot." We might be adding models to a lot of the different pieces of our organization. We probably want to reuse these in a lot of cases.

How do we make sure that we only have to build this once, but then also follow a certain pattern, so that we can scale them and monitor them, etc.? These companies have noticed that AI is here to stay to a certain degree and machine learning is probably important for them and

that they're going to be integrating into multiple different parts of their organization. I think Facebook stats said that 25% of their engineers interact with FBLearner Flow, I believe that's the accurate statistic.

If you start thinking about the size of the amount of people who will going to be interacting with a serving side of things and the fact that their machine learning and AI teams are actually fairly small compared to the rest of the developer organization, now you suddenly start seeing why it's really important to start a certain level of standardization, but also centralization around these services and the mechanisms that you do it. That's really the driving force.

This is very much in-line with what we see in terms of behavior, where not only are people using it as their hosting service, but they're also adding many more models under their accounts and different parts of their engineering organizations are using that, and that's also led and to are essentially full-blown enterprise product, where we've actually grabbed everything that we've built for the marketplace, including all our admin tools, metrics, everything that we built from the backend to run algorithmia.com and we've productized it and we now deliver it to the largest organizations in the world, so that they can actually run internally their version of a serving platform, in the similar way that Twitter built their own and Google built their own and Facebook built their own. A lot of companies do not want to build their own. They want to buy this product and that's what we've actually been giving them.

[SPONSOR MESSAGE]

[0:18:54.7] JM: The Casper mattress was designed by an in-house team of engineers that spent thousands of hours developing the mattress. As a software engineer, you know what kind of development and dedication it takes to build a great product.

The result is an exceptional product and when you put in the amount of work and effort that went into the Casper mattress, you get something that you'd use and recommend to your friends. You deserve an exceptional night's rest yourself, so that you can continue building great software.

Casper combines supportive memory foams for a sleep surface that's got just the right sync and just the right bounce. Plus, its breathable design slips cool to help you regulate your temperature through the night. Stay cool people. Stay cool.

Buying Casper mattress is completely risk-free. Casper offers free delivery and free returns with a 100-night home trial. If you don't love it, they'll pick it up and give you a full refund. Like many of the software services that we have covered on Software Engineering Daily, they are great with refunds.

Casper understands the importance of truly sleeping on a mattress before you commit, especially considering that you're going to spend a third of your life on that mattress. Amazon and Google reviews consistently rank Casper as a favorite mattress. Try it out. Get a good night's rest and upvote it yourself today.

As a special offer to Software Engineering Daily listeners, get $50 towards select mattress purchases by visiting casper.com/sedaily and using the code SEDAILY at check out, you'll get the select mattress purchases, if you go to casper.com/sedaily and enter the code SEDAILy at check out.

Thank you, Casper.

[INTERVIEW CONTINUED]

[0:21:00.9] JM: The flavors of software engineering that exist within one of these AI platforms that Uber has built, or that Facebook has built, or that Google has built, how similar are they across the different company? Because I think in our last conversation, we talked a lot about the variety of container orchestration solutions that there were two or three years ago. I mean, there still are a variety, but there's been convergence. It's one of these things where people were trying different things, but it – there's an advantage for the ecosystem as a whole to eventually converge on a single solution, because we find that many of these problems in software engineering, it's the same problem at every company, in every product.

Container orchestration is one of those examples where yeah, Mesos works in some context, Kubernetes works in some contexts, but in general, like Kubernetes is – it does the job. In our last conversation, I don't know if you remember what we talked about. The lack of a winner-take-all nature in software engineering in general, but AI training and deployment, like if I talked to Facebook about AI model training and deployment and I go and talk to Uber about AI model training and deployment, are they going to tell me that they do the same things or very similar things? Is the use case different? Is the fact that I'm developing an AI platform for Uber and is being deployed to its real-time gig economy application, is that starkly different than the problems that Facebook is tackling?

[0:22:27.8] DO: We can separate in a couple of different things; one, which is on tooling and idea backing tooling, or what they've used, and then the second one is around use cases, right? Not hard to imagine that Facebook probably spends really, really a lot of time on NLP problems, the natural language processing, understanding what the updates are, understanding what news articles say, understanding what's being – how to do relevancy for their ads, understanding what conversations are about, what people are talking about, what the relationships between people talk about.

You can see them spending a lot of time, and if you go look at their research, which they make pretty public, like they spend the immense amount of time on natural language processing. Same thing for images; they spend a ton of time on images and being able to understand the meaning of images. One of the things that if you want to group this into a category, which is a really broad category so it's a little bit unfair, but Facebook spends a lot of time doing AI for unstructured data, right? If you think about freeform text, freeform images, freeform voice to a certain degree now, so they spend a really lot of time.

You can see that they will probably, everything that they do, my intuition would be that they probably need to build the best tooling for dealing with unstructured data. From the flipside of that, Uber doesn't deal with that much unstructured data. We can go find use cases. I'm sure of them trying to figure out things on images, or maybe then further like your self-driving car stuff, but the truth is that probably most of the data that Uber deals with is structured, right? In the sense that they have GPS coordinates, movements, what the counts are saying, where they started, where the ended.

A lot of the training and a lot of the building of models is probably around route optimization, it's probably around pairing and auctions and smart agent resolution. I think they just mentioned an article recently that they're trying to detect if somebody's drunk before they get on the car or not, and they do that by understanding you call the Uber and they're picking up on signals around what your GPS movement is, how you're moving etc., etc., which sounds really creepy, but they're trying to figure out this is research, not something that they're doing. They can make an assumption of well, if you stumbled across 17 streets before you got to your Uber, then maybe you're going to be in certain state of mind that is not ideal for the driver.

The problem sets are different. Because the problem sets are different, the preferred tooling might be different. That's what one angle, right? I actually know that Facebook does a lot of PyTorch, right? PyTorch has really been, and they talked a lot about that in the cafes, the other one that they use a lot, café too, so because it works on mobile. They use certain frameworks and tools, because they're super comfortable with them. They've also done their bet on putting back work into those open-source projects to make it even better for them.

I know that Uber for example is pretty heavily invested in Tensorflow. What ends up happening is that these companies, they come especially because they're fairly research-driven, right? You have something like really famous people working on the labs they're on and all of these in terms of AI and machine learning. They make a bet around the tool and then they go build out the tooling that they need for that tool and framework.

That's a little bit different from how it's going to happen wider, right? When you start thinking about, I'm a fortune 100 company, I'll pick a random example of a company that I know has a lot of developers. JPMorgan Chase has 28,000 developers, right? Just to give you an idea, right? That's if you don't count the drivers, I'm pretty sure that's significantly more than the total amount of people that work for Uber. That's just developers, right? If you start thinking about well, if the world the machine learning is really going to penetrate every single side of development, it's going to appear in all sorts of lines of businesses, then it's not hard to start imagining that a company like JPMorgan Chase needs to start thinking, how are 28,000 developers going to interact with data science and machine learning?

At that point, the use cases are way more varied, right? You can imagine fraud on credit card, all the way to HR problems, to accounting problem, like the whole gamut of things. Where you're not going to make a bet on a tool, you're now going to have to make it a lot more generic, right? This idea behind how app servers were pretty generic, you could launch any app from them. That movement, I think as we see larger companies with larger developer populations which have much more heterogeneous use cases and needs, then more generic platforms are going to be needed and now you start building for as a general platform, it becomes a lot harder to build that type of stuff internally.

[0:27:30.4] JM: Yeah. Now when you're talking about the differences between a company like Uber and a company like Facebook, first of all these companies have the resources to build their own specific thing, and they've developed a core competency in machine learning, so they're comfortable enough rolling their own. That's one side of this conversation. The other side of it is what you said about the core difference being the structured versus unstructured data, that's more of a question of – that seems like more of a question of how is the data being presented to the training platform, or the training process?

It has less to do with the deployment, because once you have the model trained and you're ready to deploy it and present it to anybody that's going to consume it, you can separate that as a disjoint problem. Are you thinking, like is yours – is the solution that you build, is it mostly for the deployment process, or are you also involved in the training process?

[0:28:30.5] DO: You can train on our platform, though it's not what we really specialize on. We are deployment and then the runtime. To your point – I'll clarify that you're absolutely right, where the unstructured versus structured data is really important at the training time, but actually at the prediction, or classification time is just as important, because the input into the classification problem is going to be either a unstructured image, or an unstructured video, or it's going to be an actual structured data source like maybe a redshift storage.

You will see the separation between structured and unstructured data on both sides, because you're going to have to do it. I'll give you an idea. You're an e-commerce website and you're running a recommendation model on your front page right and you're tuning this recommendation model in one way, or another. The access to those indexes to the doing the

nearest neighbor type stuff, that is going to be from a structured data source in almost all cases, versus if you're –

[0:29:29.1] JM: Like a product catalog.

[0:29:30.1] DO: Correct, exactly. Product catalog, or some actually generated tables of affinity that might have been built by your data science team internally saying, “Hey, people who want this product also want this, or these products are related and stuff like that.” Versus I upload an image to Facebook and it immediately tags my friends and says, “Hey, should I tag Jeff in this? Should I tag Diego on this?” There's a different at the classification point, which is along the structured and unstructured data point.

[0:30:01.9] JM: Just to clarify for people who don't know. The difference between structured and unstructured data I think could be defined as the degree of schema – I mean, well it's hard. I mean, first of all, it's a blurry definition. Structured data would be something like, we could say structured data something like a product catalog, where the fields are well-defined. You've got the title of a product, you've got maybe the SKU number, maybe you've got an image associated with the product.

If you're talking about an image itself like a photo, a photo is a very high-dimensional piece of data. You can derive a lot of different things from a photo, something like an audio file. It's a very rich piece of data. From your point of view, why is it in terms of building a tool for people to deploy their models easily, why is the differentiation between the structured data consumption type and the unstructured data consumption type, why is that relevant to you for your – the AI deployment layer?

[0:31:02.7] DO: Absolutely. That's a great question actually. I think your definition is right on. I do also think it can it can be blurry. If you're going to be a machine learning platform that runs classification, prediction, inference, whatever you want to call it, what are the core tenants that you need to be able to do? One of them is I need to be able to retrieve data from somewhere, I need to be able to do something with it, the classification process and then I need to put the results somewhere, right? I mean, that's core. Otherwise, how does it get consumed from a application perspective.

The core difference from a platform like ours is going to be how do you access these data sources? Because accessing a database and reading from a database and processing from a database and then putting it somewhere else is different from accessing and reading from a file storage. That's the base level of why you need to differentiate. There's a theta connector issue here where you have to build out. Then how you query those connectors and how you use them and then how you move data through the system, right?

It'd be really foolish to try to move a bunch of data off of a Hadoop cluster, right? To process them elsewhere and then bring it back to the Hadoop cluster, right? You have to be careful, because in some cases, these data sets might be giant. Fom an inference and from a platform, you have to start thinking about data gravity, you have to think about where you're actually processing the data, how you're doing it, how you're connecting it and what the actual best way to connect to those data sources is.

These are really, really relevant from a production platform is understanding where – as I said, our goal as a company and our product is, I tell this to our engineering team all the time, we should be able to lift data from anywhere, process it and put it anywhere else. The actual differences between structured and unstructured data become really core to how the product interacts with the data sources at these companies. To add to that, one thing that I think is particularly relevant is that we don't get to be prescriptive about how the data works, or where the data is, or where it's being stored, because if you think about – if you're a deployment system, you're a production system, the data systems have already been created.

In many cases, if you actually think about the general process of there's data collection, data structuring, data modeling and then production and this is follows almost any one of the analytical property – any analytical process, everything from BI to traditional, predictive analytics to machine learning, these deployments at the end, right? This means that data collection was already solved to a certain degree, ETL processes already exist, there's a structuring component that already exists there, there's probably a training component either that be on the data scientist’s laptop, or in a training system, or whatever that is.

We're coming at the end, which that's exactly where we want to be, but all these other components are already there. We actually have to adapt to make sure that we work with what already is there, because nobody's going to adopt our platform if we're trying to tell them, “Hey, you should change the way you do things for the first three parts of this process to just be able to use us.”

A big part of what we do around our thinking is how do we get to be plug-and-play into the normal workflow of developers and be essentially as valuable as possible without having to influence any of the things around us?

[0:34:29.1] JM: I did a show a while ago with a company called Dremio. At Dremio, they're really working on trying to make data access within a large organization with a lot of disparate data sources a little bit easier. In that recent show that I did with Tomer from Dremio, we talked a lot about how data is stored at a large company like Coca-Cola, or JPMorgan with its 28,000 developers, or a Procter & Gamble, or State Farm. One of these older organizations that did not start out as a software company, but they have had to become a software company and they have such a high volume of data. There's so much value trapped in that data.

There's a lot of problems that are trying to be solved – those companies are trying to solve. There's a lot of vendors that are trying to build the right solutions to give to those companies. I'm sure you're starting to look at this at this area, because you're trying to solve one specific problem that these kinds of enterprises are going to encounter. Can you give me a picture for the data problems within one of these big enterprises and how you're hoping to address those problems?

[0:35:45.9] DO: Yes. Again, there's a ton of companies that are focused on the data problem, data management, data structuring, collection, and this is really something that we don't – we can have an opinion and we obviously help drive an opinion around certain things, but we pretty much don't get involved. We really have made our platform plug and play with almost any data management provider out there, so that once this issue is solved for them, all we need to be is like, “Okay, what's your API to access these systems?” Then we can pull in – we’ll be able to pull from there or push to there with the security and all that stuff that you do.

We try to be really, really not prescriptive on this because I think it's an entire industry to begin with. It would be impossible for a company to focus on multiple of these things. When you're going to look at these companies from that perspective to your original point is that you have to just go under the assumption that there's going to be a level of heterogeneous data storage there, and that I'm sure that somebody would love to be we store everything and these three types of data stores would be great, but that's never going to happen in a large organization. That only works maybe in a startup, and that's beyond that. It doesn't scale.

Then you start thinking about, “Okay, well what's the data management process? What's the modernization?” It gets exasperated even more. If you think about a modern organization today, any of the ones that you mentioned, now you're starting to say, “Hey, not only is it that there's dozens of different kinds of storage for different things, different use cases, different groups, but they're actually stored in different locations.” Not only in different locations, sometimes in completely different types of infrastructure.

You might have a team that is starting some stuff on Azure, you might have a team that’s starting some stuff on AWS. You might have really, really large data centers built out that are more traditional IBM, Oracle systems. An organization that is looking and saying, well hey, we need to be able to access this data from anywhere and at any time and to build up the future of our machine learning systems is now thinking, well, I need to access all of these, how do I actually make this available under one plane?

That is actually something that there's companies out there solving, right? There's companies that look at that problem as a whole, and I provide that in companies like Tamer and Muda are some of the ones that I’m familiar with. They’re a 100% focused on the security and access and trying to provide a homogeneous front to all these other systems that need to query data, but I think the core is it's a huge problem, right? I think one of our large fortune 100 customers, I had a conversation with their CIO and they were saying they had 4,000 SQL, like a start – 4,000 PostgresSQL databases 4,000. 4,000 right?

Again, maybe that's an inefficiency, maybe they only need 2,000. I don't know. I don't know enough about their business to be able to even give an opinion around that. They are the experts, but that's the general – so these are the problems that they deal with. There's a whole

industry of data management that provides the front you access and access controls. When we come in as a production and deployment platform, our rule is we need to work with that. There's no way around it. We just need to work with those.

The good thing is is that those companies have spent a ton of time and work on making their front-end, right? Like API front end, or their access front end pretty universal, right? They've done it through REST API, they've done it through a generic JDBC connectors. We can very easily plug into that and feed off of it. Our whole point is that we start after that system has already been either developed or made present.

[0:39:46.4] JM: If I understand it correctly, if I'm an engineer at a big insurance company and I've just been hired to improve the insurance rate risk calculator that is based on the data that's in these legacy data stores, I'm sitting down to build my machine learning model, I've already got the data connector problem probably solved for me, and otherwise, why would I have been hired to do my machine learning job? I'm sitting now to do my machine learning job, I connect to the data connectors, I start training a model and the model gets good enough, I've trained it and it works on my machine, and then I get to the point where I want to make it an API endpoint so that biginsurancecompany.com can access my machine learning model as an API endpoint.

This is where, this is the point at which plenty of developers hit a friction, because there's all of these different things that go along with the deployment, like can you deploy in different languages? Or are things getting cashed appropriately? Are you running on a performance processor? Can you version your model? If you get a lot of large influx of data and you want to build a new version of the model, but you don't want to maybe deploy it to every customer at once. Maybe you want to do some AB testing or something like that, maybe you want to monitor and log your machine learning model deployment?

The point here is that there are specific deployment issues for machine learning models that are probably not captured by other continuous integration systems and that's the problem that you are solving specifically.

[0:41:25.7] DO: Yes. I think you described that really well. I do want to make one asterisk, which is – I have to do a little bit of a shout out to some unsung hero. You said, hey, they get hired as a

scientist and all of this is solved for them. Ideally, yes. That's how it should be, right? The fact is that a lot of these data science and machine learning heads and VPs of analytics are being brought in with none of this being solved, which essentially puts them in a role where they need to actually go to solve the whole data management, and did like process out.

The ideal world would be and know this is what we advise companies is that you can't really start with any machine learning, real machine learning projects and data science projects until you've at least had this addressed to a certain degree. There's plenty of as I said, unsung heroes there who are being brought into these organizations, who they actually have to go gorilla warrior their way through every single one of those problems, before they can even get started on what they got hired to do. I do want to just point that out, because there's – I know you have a lot of – you probably have a lot of listeners in the space and they're probably thinking –

[0:42:30.6] JM: They're doing data cleaning.

[0:42:31.5] DO: Yeah.

[0:42:32.2] JM: I just told them that they're not doing data cleaning. They're thinking, "No, my job is data cleaning."

[0:42:36.5] DO: Yeah, exactly. You know that they're sitting there being like, "Well yeah, gosh. That would be great if that was actually how it worked." There's a lot of unsung heroes there who are fighting the battle all the way up. If you're doing and design the process, the expectation and the success, and you actually can correlate quite easily companies that have been successful in their data science and machine learning projects to having a very serious and dedicated data strategy for the structuring and the modeling, and once that is available, truly bringing in these really smart folks who can grab that data and iterate through it and build it. Then they want to productionize.

Then absolutely, that's where we solve the issue of scale and uptime and the API part and moving it around, moving your data around and scaling it. Then connecting multiple models together, right? A big part that people don't think is that even in the world of model deployment,

there's pre-processing functions. The classification functions might be 1, 2, 3, it's an ensemble, there might be a post-processing function. All that pipeline needs to be scaled together, it needs to be secured, it needs to be potentially audited if you're in a regulated industry.

As you said, the full CICD process, plus the actual uptime is really where the problem becomes really apparent and where they need a solution. I go to – like a joke is usually saying like, “Look, anybody can run a web server. It's actually really, really easy to lift up a red server, or web server, but not everybody can run Reddit, right?” That's where the difference is, right? Anybody could stand up a model with a simple API and run it on their laptop, but if you're actually going to depend on the system to run your business, if you're going to depend on the system to run at the scale and potentially either business-critical, or in some cases if you think about government like life, in situations where it might be life and death involved, you better have security and audit and uptime and the things that you need around that.

[SPONSOR MESSAGE]

[0:44:49.6] JM: A thank you to our sponsor Datadog; a cloud monitoring platform bringing full visibility to dynamic infrastructure and applications. Create beautiful dashboards, set powerful machine learning based alerts and collaborate with your team to resolve performance issues. You can start a free trial today and get a free t-shirt from Datadog by going to softwareengineeringdaily.com/datadog.

Datadog integrates seamlessly with more than 200 technologies, including Google cloud platform, AWS, Docker, PagerDuty and Slack. With fast installation and setup, plus APIs and open source libraries for custom instrumentation, Datadog makes it easy for teams to monitor every layer of their stack in one place.

Don't take our word for it. You can start a free trial today and Datadog will send you a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started. Thank you too Datadog.

[INTERVIEW CONTINUED]

[0:45:57.5] JM: What's the conversation like with these enterprises? As you've gone from the world of building a marketplace to building – I mean, to selling to the enterprise, it's a very different world to exist in. How have you adapted your ability of selling, I guess, or what has that evolution looked like from a business point of view?

[0:46:22.5] DO: Yeah. I take probably the two most important things are two Ps. Patience and partnership. Especially early on, really thinking through how you and this enterprise, how do you bring value to them immediately? Then how do you work over a longer period of time to make sure that you continue adding value, or increasing value. If you can actually establish that partnership relationship with some of these larger, these long enterprises. On top of that, you have the patience, right? Because they go hand in hand, the results are really, really good.

It's not a secret some of these enterprise deals can be like king-making to a organization from a financial perspective, also from a reputation perspective. For us, it's really been – we started really early on, as soon as 2016 is when we started working with enterprises. We kept it for the most part on the down-low and we did – and it was a big part of exploration and partnership and holding hand in hand and saying, “Okay, what do you need? What are the problems that you need? How does our platform solve them? How can we make ourselves more enterprise-ready?”

Then the final part really is hiring people that have done this before. A little bit of, I wouldn’t say [inaudible 0:47:42.4]. We've added some personnel that is very experienced in terms of working with enterprises and then holding to our first two things, which is I said the patience and the partnership part. We treat our large enterprises not as customers, but we treat them as partners where we understand that our product being at the forefront of this world, of this world of machine learning, like it needs to be constantly evolving and there's no better people to help us evolve it than the ones with the actual problem that we’re solving. That mentality of partnership has gotten us really far.

[0:48:21.2] JM: From an engineering point of view, when I look at this as a pure engineering problem, it sounds you're building a Heroku for machine learning models. You're really trying to give a solution where you take the problems out of the purview of the developer and that's what Heroku did in contrast to something like EC2. How have you changed the platform itself?

Because the platform before was I think more just about this this marketplace side of things, but I assume you've had to build more tooling that is internal, or get a logged in experience as opposed to this marketplace deployment experience. How has the product tooling changed?

[0:49:09.5] DO: Interesting enough that it hasn't. I'll explain exactly why. We had to run this thing, like we as Algorithmia, we had to run with over 70,000 developers over 5,000 models, 24/7 on-call, SLAs to large customers who are using our platform. We have to build all of these in what we – with a small team and being a startup, it's always been automation, automation, automation, automation.

One of the things, if you think about it, we built a building system for the marketplace, we built a audit system for security, we built the security that we needed, because we run random code from the internet. There's a bunch of things that we built as tools for ourselves, which ended up translating into exactly the features that enterprises wanted to be able to learn this themselves. At the end of the day, what they're looking for, if you think about maybe there's no transaction system, but the running internal marketplaces, right? There's teams that are making available models inside these organizations and there's other teams that want to consume them.

The actual interaction between the consumption and the creation is actually identical, even to the state that the actual people who are actually involved, they're just all under one roof in their organization. The tooling perspective actually has an – we obviously advanced it, but a really interesting part is that the actual philosophy around how to do this has been how are we going to enable a really small team inside these organizations to run a massive production system for machine learning across the organization? Happens to be that we've been running a massive production system for the world and that we built tooling for ourselves, these things actually translate really well.

There's obviously other things that – certain features, but I would say that if you think about single sign-on and off controls and being able to generate logs into an enterprise logging system. There's these extra enterprise features that we've actually have to go build as well, but the core from an engineering perspective hasn't changed at all, which is really fascinating and exciting and actually why we've been able to ramp up so quickly into this side of the business.

[0:51:28.9] JM: Last time we talked, we had some conversation about the homegrown containerization and compute management platform that you've built. Have you done any re-platforming on the internal engineering side?

[0:51:43.0] DO: At the core of it, our platform is git, Docker, Kubernetes. Obviously, a bunch of complexity internally to that and a bunch of things that we've built, but that's the core components of our platform and that hasn't changed. We've already advanced it. It's been fascinating to see Kubernetes advance at the speed that it's advancing, all great stuff for us. We have a general philosophy that if there's an open source project that we can use and contribute to, then and we don't need a homegrown solution, then that's the way to go.

We're constantly evaluating components of our platform and saying, "Hey, can we just use something that already is out there?" That's how I would – that's how we've approached it. There's been a lot of changes in terms of the prompt inside those projects, right? Last time we talked, Kubernetes had almost none, or very little GPU support, which is core-core to what we do. We run so many GPUs.

Now Kubernetes is actually really a lot better. It's not perfect, it's not exactly where we want it to be, but we've been able to actually start paring down certain pieces of our custom code to be able to take advantage of more native tooling inside Kubernetes. That has allowed us to accelerate, so that we can invest a little heavier in other areas that are really not in the Kubernetes world.

[0:53:12.4] JM: Do you manage your own Kubernetes, or do you use one of these hosted managed Kubernetes providers?

[0:53:20.1] DO: We manage and build our own. Today, we manage our own on a couple of different clouds. We also work cross-cloud, which is another thing that just probably really about it, so we wouldn't be able to use one managed service to jump to the other managed service, although that sounds awesome. If we can get there one day, that would be great. We build out our own Kubernetes from scratch and we go and deploy and when we run ourselves.

[0:53:47.4] JM: Is that cross cloud important to you from a failover perspective, or because some of your customers are on AWS, some of your customers are on Google, some are on Azure and you want to be able to locate the models close to their other infrastructure?

[0:54:04.6] DO: The latter is really the way that we look at it. Data sources, data gravity is really important. If you think about data is expensive and heavy to move, that compute is super light and cheap. Especially with containerization and with things like terraform and scripting, we can go build out very, very quickly compute resources in different platforms. Moving data tends to be the slower process. If you have a database in Azure, we're going to try to go build out our infrastructure in Azure to build that, continue to process that.

If your data is somewhere in AWS, or an S3, we're going to want to build it up there. Then on top of that, on a per region basis, right? Because you don't want to pay the data transfer in between regions. These are all things that when you start thinking production, while in training you're just like, “Well, the data is here, we're just going to build it here and then we're going to tear the whole thing down.” When you're actually doing in production, like the data actually be in multiple different places, or the model is the one that actually needs to move from region to region based on where the data sources are and stuff like that.

These are the types of problems that they only start being thought about once, hey, my model works on my laptop. What's next? Then they start realizing that there's this world of considerations that need to be taken, which is really where we specialize in.

[0:55:28.2] JM: When we last talked, this is when I was still living in in Seattle, in the Seattle area. I thought algorithm is really cool, because I think it's one of the most interesting startups in the Seattle area. It was fascinating stopping by the office and talking to some people, talking to yourself. I'm not totally unsurprised to see that the business has continued to succeed and you've found other verticals to get into. It sounds like things are going really well and it sounds like a fun and interesting job, because there's a lot of opportunities for growth of course and other businesses to be built on top of it. What have been some of the challenges that you've encountered?

[0:56:06.7] DO: I think any startup is going to have growing pains, right? Going from eight people to 16 to 27 to jumping beyond that, at every single jump there is challenges, from a company perspective on how do you keep on mission for adding management layer, having people really rally around things and do that. I think there's the general growing pains of a startup. I mean, that said, it's still a super exciting place to work. I think, that's probably the most exciting part of it all, which is we've been able to bring on a lot of really, really amazing teammates to continue this mission and grow it, and it's accelerated everything.

When you accelerate everything, you also hit potholes a little bit harder, right? That's natural. How quickly, how hard can you hit that pothole without popping the tires? It's something that a startup is always testing, I feel like. From a challenging perspective, I would say your growth is hard. Being able to define a new category, which is what we're doing, like when we lost talked, even though we were doing machine learning deployment if you really think about it, nobody was talking about it. This is not a thing that existed, not because most people haven't gotten to that side of the problem yet.

Now it's actually a conversation and you hear a lot about it. You hear more companies popping up around it, you hear about different cloud providers thinking about how they are going to solve this problem. That when we're essentially creating a new category, or you're in a space where this doesn't exist, but so now you have to actually educate your customer on why you're needed and then it's really obvious to them at that point, but then educate the rest of the organizations and the world on like, "Hey, this is when you need us, this isn't when you don't, this is what we do, this is what we don't," that's really hard. That's a really hard problem to solve. That's been another one of the places that we've spend a lot of cycles around messaging and how we change things and stuff like that.

Even this conversation with you just been fantastic, if you think about it, like our product hasn't changed that much. I mean, it's gotten better, but the way that we have to talk about it, the problem-solving, like even you picked up on it and said, "Hey, this is how it shifted. It's almost like a new vertical." That is actually a challenge in itself, right? How do I address that? How do you educate yourself, everybody in the team, the organizations you're interacting with and then the world at large.

[0:58:50.6] JM: Diego Oppenheimer, thanks for coming back on Software Engineering Daily, it's been really great talking to you.

[0:58:54.4] DO: It's been my pleasure. Thank you so much for having me.

[END OF INTERVIEW]

[0:58:58.8] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]