EPISODE 766

[INTRODUCTION]

[00:00:00] JM: Data infrastructure is advancing beyond the days of Hadoop MapReduce single node databases and nightly reporting. Companies are adopting modern data warehouses, streaming data systems and cloud specific data tools like BigQuery. Every company with a large amount of data wants to aggregate that data into a data lake and make the data available to developers. All of these data can be used to power machine learning models, which can potentially improve every area within a company where they have historical data.

Data pipeline is a term used to describe the process of preparing data, building machine learning models, deploying those models and tracking the results of those models. Pachyderm is a company and open source project that is focused on deployment, management and scalability of data pipelines. Pachyderm allows developers to version data, track the state of datasets, back test machine learning models and collaborate on data. It also tackles the very hard problem of machine learning auditability.

Joe Doliner is the CEO of Pachyderm and he returns to the show to discuss his experience building Pachyderm over the last five years. Data infrastructure has changed a lot in five years and the world has moved in a direction that has benefit Pachyderm with more infrastructure moving to containers and more data teams advancing beyond a world of just Hadoop MapReduce. In today's show, Joe talks about modern infrastructure, data provenance and the long-term vision of Pachyderm.

[SPONSOR MESSAGE]

[00:01:43] JM: STELLARES is a job recommendation engine for software engineers. STELLARES finds career opportunities that fit you perfectly by taking all your preferences into account. The STELLARES process starts with a conversation. You text with a conversational interface and the chat bot helps you narrow down what you are interested in. Then after you have given STELLARES your preferences, STELLARES uses its machine learning algorithms to factor in the subtle aspects of the job search so that you find your perfect job, from salary, to work-life balance, to team fit and personal learning goals.

To find out more about STELLARES, go to stellares.ai/sedaily. That's stellares.ai/sedaily, S-T-E-L-L-A-R-E-S.ai/sedaily, and after you go through the job matching process, STELLARES gives you a warm introduction to the engineering teams that you're interested in so there's never any pressure, just opportunities to explore what's out there. Check out stellares.ai/sedaily and find a new way to look for a job.

Thanks to STELLARES for supporting Software Engineering Daily.

[INTERVIEW]

[00:03:11] JM: Joe Doliner, you are the CEO of Pachyderm. Welcome back to Software Engineering Daily.

[00:03:16] JD: Thank you, Jeff. It's great to be here again.

[00:03:18] JM: We last spoke in 2015. How has data infrastructure changed in the last four years?

[00:03:25] JD: Well, I think more than just data infrastructure, changing the general expectations of data have changed. If you think back to 2015, there was really none of this worry about where is my data going? I mean, there were still like data hacks and people would be upset about that, but there wasn't the same level of like firms getting data and then targeting political ads, people like the Equifax breach hadn't happened and things like that.

So, for us, it really felt as if people like woke up over the last maybe 3 to 4 years and realize that data was this really important thing that was affecting like all the way down the consumer lives and that they needed to have a lot stronger guarantees in place of where their data was going. There was no GDPR then. I think the GDPR was like sort of in the process, but now it's actually in effect, things like that. So that's what’s changed in terms of what people are asking for for data.

In terms of what's changed in terms of the actual infrastructure, probably the biggest change has just been how much ML is taken off and how much the infrastructure has had to move to support that. So basically every data product that we know about, and we're of course trying to keep tabs on all of them, it has been moving toward having the best story possible for machine learning, for running things like TensorFlow, and PyTorch and Keras and things like that. That seems to be the two major themes that I've seen.

[00:04:47] JM: Does this apply to startups and enterprises? Because I've talked to plenty of startups who are updating their data infrastructure, they're making their data infrastructure more accessible to data scientists and machine learning engineers. These are companies like DoorDash, or Uber, or Airbnb. I feel like at the level of the enterprise, many of them are still working on microservices, and on the data side of things, it's more about they're just trying to get their data under control. They're trying to figure out what is our data lake. They're just trying to make their data available and they haven't moved on yet to actually getting machine learning models built and deployed. To what degree is that accurate?

[00:05:36] JD: I think it really depends on the company. We talk to companies sort of up and down the like tiny little startup to gigantic enterprise company stack, and we see some small startups whose entire company is based around being able to do something in machine learning really, really well. So then they're just laser-focused on building that competency.

We also see some small companies that are like, "Look, we need to have some data infrastructure, but it's simple stuff. It's like SQL tables. We can just use like Google Bigtable or something like that." Wait, not BigTable – BigQuery, for something like that. Then on enterprise side, we do see a lot of what you're talking about of companies just saying, "We need some – To bring some sense to this chaos. We need to be able to understand our data and be able to see it," but we also see a lot of companies that have basically this edict from above that's saying like, "We need to have a five-year timeline for like how are we going to have machine learning touch every single aspect of our business," and you get very different approaches to that. Like sometimes those types of projects are kind of boondoggle projects where like someone heard the machine learning was cool, they want to apply to their company and so they're just like drop a bunch of money on all of these different software stacks to try to do something.

Then you see some companies where it actually is becoming just like core to what they do. So I think that this is for e-commerce businesses, where like recommendations and understanding user profiles. Those are some of the most sophisticated shops I've seen in terms of actually needing to use machine learning to do something. That's one of the things that I think you'll see in any like nascent technology, is you've got the people who are really laser-focused on actually getting the real value out of it, and then as the hype builds on it, you've got more and more people who just sort of want to dip their foot in the pool, because it's cool, and sometimes long-term, those turn into really interesting projects, but a lot of times they don't really go anywhere.

[00:07:28] JM: If I am an enterprise, I've got some ETL jobs that are moving data from one place to another. I've got some nightly reporting jobs that are running. How do I need to update my data infrastructure to get from that world to the world where I can have machine learning models?

[00:07:49] JD: It really depends on what you want to do with those machine learning models. If you just want to have a machine learning model built on one of those, then you can just hire a data scientist, allow him or her to download that dataset and you train those models, maybe provide them a box with some GPU's or something like that, and that's often the first step that companies go through.

The difficulty is that there's a big gap between that and having machine learning models that are getting trained every single night based on new data are getting deployed to your servers and are becoming like part of your infrastructure and part of whatever you're offering is, be it a search engine, e-commerce website or something like that. Most importantly, having auditability in all of these so that you can say like, "Oh! We've got this deployed on the site. We got a bunch of fraud today. Let's check the fraud model. Let's check the data that went into it." So those are sort of the growing pains that we always see companies go through, right?

So it's unfortunately very hard to say and a lot of companies ask us exactly that question of like, "We need to be doing machine learning. How do we do some machine learning?" and we always have to say, "Well, you can do machine learning on your laptop right now, but that's not actually what you mean by machine learning. So you're going to have to tell us more exactly what it is you mean.”

[00:09:00] JM: Your company, Pachyderm, is built around the data pipeline process. Can you explain what you mean by the war with the words data pipeline?

[00:09:13] JD: Yeah, absolutely, and this is a really good thing to clarify, because the word pipeline in general in computing has gotten really, really overloaded. So when we talk about data pipelines, we basically mean that there's this swath of data sitting in object storage generally referred to as the data lake, and it's called the data lake because data is constantly flowing into it and data can be sort of taken out of it as well. When you have a pipeline, then that means that as data flows into your data lake, it automatically gets processed through some set of steps that you have specified ahead of time and data comes out the other end in some sort of a process form that’s going to be useful to you, and that could be formatted stuff that you can load into BigQuery so that people can do SQL queries on it. That could be a machine learning model that gets pushed on to your website to be served somewhere. So that's the data pipelining functionality that Pachyderm provides, is the ability to do that.

The other thing that we sort of provide on top of that – Well, below that, we provide the actual data storage, that data lake, but then on top of that we provide all sorts of auditability, where you can see like this individual piece of data flowed in on this day and it flowed all the way through the pipeline and here's what came out the other end, and we have version control of all of these, the input data and the output data. So you can actually rollback those results to a previous past result if you don't like that new data that flowed in. You can do reverts and things like that.

[00:10:40] JM: Your product has evolved over the last four years and you were able to evolve because there were a number of market trends that you were correct about, like one big trend that you got right is the need for new data engineering products. You saw that the data engineering – Four years ago we’re talking about replacing Hadoop and you were right about that. Hadoop has its issues. Hadoop MapReduce has its issues in terms of how it serves the kinds of applications that we want to build on top of it. So that was one big trend. Another was the shift towards containers for data science. Are there any other big trends when you think back over the last four or five years that have really helped you?

[00:11:25] JD: I think that the other big trends that have helped us are the need for data provenance and auditability. It's been this kind of funny journey for that feature set and that idea within our product, because it started is basically something that we needed to do to get the model to work with version control. It was almost an implementation detail. Then we sort of, as we're looking at it, like, "Okay. What is the name of this – What would we call this information about where data came from?" and we're like, "Oh! Well, people call this provenance," and actually this is like a user-facing feature that people were talking about. They were just talking about it like a little bit. Like just some people were interested in that in like 2015 when we first added it, and we thought of it as this feature that was just useful for data scientists to do their jobs, right? Because anytime you're looking at some sort of results and you say, "Oh, this looks a little bit weird, or maybe this looks really good," being able to just track that back and reproduce your results is just a core functionality that you want to do any kind of science.

After that, there was this whole movement toward users wanting to be able to see where their data came from, and then the GDPR actually added like a legal requirement that you're always able to give people an explanation basically for anything that any sort of an algorithmic decision that you make if you're a bank, for example, making a decision on a loan. You need to be able to say, "This is clean data. It doesn't include your race. It doesn't include your gender. Here's what it includes."

So that's one of the big trends that I feel as if we actually got really lucky on. When you talk about the container trend and the data infrastructure trend, that was a very clear, like I saw that and I was like, "Okay. This is going here and that's where I want to go." With provenance, I was just like, "Okay. This seems important right now. So we're going to do this," and then we were kind of surprised at all of the other stuff that came from that.

The other big trend that I think we've managed to latch on to is machine learning, and that's by far we've made the least – We haven't been exactly early to that game more so than anybody else. I think everybody sort of saw that machine learning and once like neural net started getting such amazing results that that was going to be the most important data use case for at least the next five years, probably the next 10 years. So we've basically done everything we can on that to integrate with the best platforms out there, and we got sort of lucky with the platforms that we chose. TensorFlow is probably the leader in terms of what people want to use to do machine learning right now, and because that's a Google and Kubernetes is from Google, there's Kubeflow, which is one of the main ways that people are deploying it on to infrastructure right now, and that's something that just works really, really nicely for us, because we already have everything on Kubernetes. So that's sort of like additional dividends I think from having picked the right just deployment and container ecosystem to be working in.

[00:14:13] JM: Okay, this is great timing, because we just did a show about Kubeflow. It's actually going to be published tomorrow, I think, and as I was doing a show about that I was like, "Wow! I'm surprised there's not like a Kubeflow company yet." IT sounds like you actually backed into – I mean, I realized Kubeflow is a nascent product. Maybe it's not a company that you can build a product around, but it certainly sounds like you are in the right place at the right time regarding that project.

[00:14:40] JD: Yeah. Yeah, we definitely were, and we just got to sort of talk to the right people at the right time. We met them at Kubecon. I remember talking to the Kubeflow guys when I'm pretty sure they described it as there was half an engineer at Google working on this and a project manager. Like, "I'm working on this, but it's only half of what I'm doing. I've got this whole other thing," and I actually don't know what the state of Kubeflow internally at Google is right now, but I'm pretty sure based on what I'm seeing outwardly that it's more than half an engineer at this point.

Yeah, we sort of just backed into that. People started asking us, "Hey, can we run Kubeflow on Pachyderm," and it's very easy to do that, because everything is running on top of Kubernetes and we are sort of trying to push the limits of that and make it as ergonomic and just a first-class citizen within Pachyderm as possible, which again is just pretty easy to do, because it's all Kubernetes and it's all containers.

[00:15:34] JM: I want to explain more about what Pachyderm is. So I don't know how well I even understand the product. I haven't used it firsthand. What I know about orchestrating workflows for different data jobs is that people often use a product called Airflow, and you can use Airflow to pipe together these different tasks that you're doing across your cluster. Can you give some more context for what Pachyderm does and how it compares to something like Airflow?

[00:16:07] JD: Yeah! I'm actually a great guy to answer that question, because I was actually working at Airbnb when we open sourced Airflow. That's why it's got air in the name is because it's an Airbnb product.

So Airflow is basically just a pipeline scheduler, and you specify a directed acyclic graph of data tasks that you want to happen in Python, and those data tasks can be a bunch of different things. I mean, I think you can shell out the stuff. So in theory, it can be anything. But the main things that people use it for are like kicking off ETL jobs. So like running SQL style queries on Hive or Presto, or data warehouses like Redshift and – What's Google? BigQuery ones.

It's pretty tuned toward the Hadoop ecosystem, because that's all the stuff that we were using at Airbnb at the time. So it interfaces nicely with Presto and Hive, with things like MapReduce, you can use in there too with HDFS, like that.

Pachyderm has basically one half of it that is pretty comparable to Airflow, and that's the Pachyderm pipeline system. So that's the same thing. It's about scheduling a series of data tasks or it isn't as tuned to the Hadoop ecosystem. It's not really tuned to the Hadoop ecosystem at all. We're tuned toward just arbitrary free form data tasks. So that would be things like OpenCV in a container. Things like GATK, which is the genetics toolkit that biologists like to use.

Also very tuned toward machine learning pipelines and stuff like that, because you can put TensorFlow in a container. You can put [inaudible 00:17:37] in a container, and because we're scheduling containers on Kubernetes, we give you a very, very easy way to basically pass parameters to the Kubernetes such as this task needs a GPU attached to it, and then Kubernetes will make sure that it schedules it on a machine with GPU and it has access to that GPU, and then when that task is done, Kubernetes de-schedules it and the next guy can use the GPU. So it's sort of managing your resources for you. That's half of the Pachyderm product.

The other half of the Pachyderm product is the file system, and there is nothing in Airflow like that, because Airflow depends on things like HDFS, or S3 to be their underlying file system. The Pachyderm file system is a pretty thin layer on top of an object store. We support all of the major object stores, so like S3, GCS, Azure Blob Storage. What we add on top of that is version control semantics.

So the ability for you to say, “Here's a repository of data, and I want to commit all of this stuff atomically, and then that's a new commit, and then the next day I'm going to commit some more stuff, but if I want to I can roll back and I can go back to any of my previous versions,” and the reason that we wanted to build both of these components rather than just focusing on one is that they’re very, very deeply integrated with each other.

So the Pachyderm pipeline system, because it has this underlying version controlled file system, is able to know at any time what data it's already processed. So when you might have a petabyte of data sitting in the Pachyderm file system and then you add an extra gigabyte of data, and that's of course going to kick off a bunch of pipelines and they're all going to run and process stuff, but they're going to know that only a gigabyte of that data is fresh. So they’re only get a process that gigabyte and then you're going to see the results as if you’d process the full petabyte immediately.

When I was at Airbnb, we didn't have any sort of incremental processing like that, and so we would just process everything from scratch, which just means that you’re sort of paying for extra compute, which for us at Airbnb wasn't a huge deal, but it also means that you get results out a lot slower. So we basically ran things on a daily schedule at Airbnb, where data would come in during the day and then that evening at midnight the pipelines would kick off and everything would run, and the hope was that they'd all be done by 9 AM the next morning.

In Pachyderm, you can sort of always be running stuff. So if a new piece of data comes in and kicks off and starts processing it immediately and then just merges in that tiny little bit of extra data so you’re much more up-to-date.

[00:20:04] JM: If I’m a data scientist or a machine learning engineer, or for that matter, a software engineer who works at a company where maybe the machine learning engineer just quit, and like I have been put in charge of doing this data engineering stuff. I don't know anything about it. Is my experience with Pachyderm, is it on the command line? Is it a GUI? What am I doing to interact with these data pipelines and do things like data provenance?

[00:20:32] JD: We have both. So there is a command line client that's sort of the first line of defense, and for me that's the thing that I find most comfortable. We also have a GUI that gets deployed with a Pachyderm installation that's actually part of our enterprise product. So you can go into there and you can do all of the functionality from the command line. You can put data in. You can create pipelines and things like that.

Normally, in sophisticated deployments, people are using a combination of the GUI for exploratory stuff, programmatic access, so like the Go client or the Python client for the actual like putting up data, and then the command line just sort of like poke around at things and do like batch operations. Like if I want to script something and pause every single pipeline or something like that, that might be something nice to do on the command line. That's basically what the experience is. We try to sort of offer all three, because we have a very wide set of different users.

One of the other things that really sort of discriminates or users into different camps is how comfortable they are with Kubernetes, and we get sort of a full spectrum from people who are Kubernetes experts know much, much more about Kubernetes than us at Pachyderm. So they are totally comfortable with using Kubernetes and with just looking like, "Oh, this pod failed a schedule for this reason," all the way to people who are pure data scientists. They're not even really that comfortable on the command line, but they definitely don't want to know about like a complicated infrastructure product like Kubernetes.

So for them, we sort of have to offer a much like easier insight into what's going on in Kubernetes in terms that they understand, because a lot of times Kubernetes is still a fairly new piece of software and they run into things like, "Oh my EBS volume is failing to attach to this node," or something like that. Yeah, you kind of need to be a Kubernetes expert to be able to dive in there and figure it out, and so we have to help people with that a lot.

[SPONSOR MESSAGE]

[00:22:34] JM: MongoDB is the most popular, non-relational database and it is very easy to use. Whether you work at a startup or a Fortune 100 company, chances are that some team or someone within your company is using MongoDB for work and personal projects.

Now with MongoDB Stitch, you can build secure and extend your MongoDB applications easily and reliably. MongoDB Stitch is a serverless platform from MongoDB. It allows you to build rich interactions with your database.

Use Stitch triggers to react to database changes and authentication events in real-time. Automatically sync data between documents held in MongoDB mobile or in the database backend. Use Stitch functions to run functions in the cloud.

To try it out yourself today, experiment with $10 in free credit towards MongoDB's cloud products; MongoDB Atlas, the hosted MongoDB database service, and Stitch. You can get these $10 in free credits by going to mongodb.com/sedaily. Try out the MongoDB platform by going to mongodb.com/sedaily. Get $10 in free credit. It's the perfect amount to get going with that side project that you've been meaning to build.

You can try out serverless with MongoDB. Serverless is an emergent pattern. It's something that you want to get acquainted with if you're a developer, and getting that $10 in free credit to have a serverless platform right next to your Mongo database is really a great place to start. So go to mongodb.com/sedaily. Claim that credit, and thanks to MongoDB for being a sponsor of Software Engineering Daily. I love Mongo DB. I use it in most of my projects. It's just the database that I want to use.

Thanks to MongoDB.

[INTERVIEW CONTINUED]

[00:24:43] JM: If you've been to KubeCon, you've seen the presence of the platform as a service providers that are building businesses on top of Kubernetes and they're helping enterprises move to Kubernetes and giving the enterprises a nice GUI to help them install software and they’re helping them with consulting and they’re helping them be this sort of full stack big solution. These are like Mesosphere, Red Had OpenShift, Platform9, Rancher. It’s just like a big industry of different platform providers, platform consultancy plus Kubernetes thing providers, full stack-ness.

So if I'm an enterprise and I am looking at giving my dating engineers this platform that helps me manage my data pipelines, am I looking at this is an alternative to Mesosphere or to Red Hat OpenShift or is this like additive or is it an adjunct thing?

[00:25:45] JD: It’s purely additive. Basically this is an application that you install on any one of those platforms, basically. It does get this sort of like enterprise-ificaiton of Kubernetes, all the people sort of building stuff on top of it. This definitely causes us some headaches. There's a spectrum of how they work.

What we've seen so far is that there are companies that are sort of buying those types of platforms, and I'm sure there're ones out there that are doing lots of stuff with it, but a lot of the ones that we've seen are sort of like we have decided that we want to have a five-year timeline to get like Kubernetes running and everything running on Kubernetes and data as part of that. So we’ve signed this contract with Mesosphere, or Red Hat, something like that, and have this cluster here and it's got 10,000 nodes on it and nothing's running in it. So we’d to see if this can run in it. Then we go in and see if we can work within their firewalls and things like that, and there's always – OpenShift comes with all of these millions of customizations. So it's not quite the same as Vanilla Kubernetes and no two Red Hat, or no two Open Shift installations that we've seen are really exactly the same two. So it always requires a little bit of finagling on our part.

I think that there is real value in like bringing this to the enterprise. I come from the sort of like open source software hacker type of things. So like all of these enterprise stuff, I just look at it like this is just not what the type of stuff that I want to be doing. I of course recognize that there is real value in bringing this into the enterprise. I think that there’s – It just brings in a lot of friction is what I found, and I don't know that just comes with a territory, but a lot of the friction to me seems a little bit manufactured.

Like one of the main things that I always hear about OpenShift is they've got all of these sort of like security requirements for things within containers, and to give you an example of this, like it won't let code in a container run as root, which not running as root is something that sounds really good from all of our like inherited history of running things on a multiuse Linux system, but kind of the point of containers is that the code is contained. So it doesn't really matter if you run it as root. That's why every Docker container just by default runs the process as root, and there have been some security vulnerabilities because of this that have had to be fixed. So Red Hat can sort of claim, "Well, we wouldn't have these if you were running things as root within there," but I'm very dubious when I see basically like two security systems being layered on top of each other in the hopes that, well, if one doesn't catch it, the other will, because I think that what happens when you do that is like both security systems think like, "Well, we've got two security systems here. So like maybe they'll catch this one," and there ceases to be real pressure for the security systems to be airtight. I don't know.

It also can save you some real security flaws and practice. So it can go either way, but thus far my opinion of those sort of like Kubernetes as a service and things like that is that it's hard for someone to really have a value add over Google's Kubernetes service, GKE, and maybe you don't want to be on Google, and Amazon's got an equivalent one and Azure has got an equivalent one, and those ones pride themselves on being just pure Kubernetes, because that's what they think the market wants, and OpenShifts and the like pride themselves on not being pure Kubernetes, because they want to believe the pure Kubernetes is insecure and you need all of their added stuff on top get stuff to work, and maybe these or just different markets want different things. But like last time I checked, Azure isn't exactly stupid when it comes to appealing to these enterprise customers either. So I think that there's some sort of disagreement that's going to play out here on the market.

There's also for OpenShift the added wrinkle that that company just got acquired by IBM, right? You can see that as, "Well, IBM was using this as a way to get into container infrastructure, which they really care about, and so OpenShift isn't going anywhere. In fact, it's going to have more support or it's going to go through sort of like slow company acquisition death within IBM, because they're even more enterprisy than Red Hat.

So only time will tell where our policy is that we're willing to talk to people on any of these systems, and we think that without too much like where Pachyderm should be able to work and get deployed on any of these systems, we just sort of use it is a signal of like who's really serious about getting stuff done versus who's sort of has some container initiative from above where they need to make it seem like they're doing work and nothing is actually going to happen.

I mean, one kind of simple test that you can do about this is if you Google around on the internet, you can find a million different blog posts by people that are like, "Here's how we did X with Kubernetes," and it's very clearly like written by an engineer and they show your dashboards, they show your logs, like we have this gigantic Kubernetes cluster deployed. All of these stuff is running on it. It's mission critical. Like here's this outage we hit. Here's how we fixed it. I've yet to find an article like that about OpenShift, or about any of these providers.

Maybe it's because the people who are using OpenShift don't write articles like this. Maybe it's because the people who are using OpenShift when they do write articles like this, they just describe it as Kubernetes, because that's the way to latch on to the trend. It could be any of these things. It's very, very hard to know, but in my experience so far, the sense that I get, is the people are using Vanilla Kubernetes to get a lot of very, very real mission-critical work done, and OpenShift is sort of still like on a five-year timeline with the enterprise, with the big enterprises.

[00:31:21] JM: This is getting into the business side of things a little into the weeds. We just did a show about bottoms up versus top-down sales process, where like – It sounds like you're getting into the enterprise and you're having these enterprise conversations, and this bottoms up versus top-down thing for open source software like Pachyderm, one thing I've heard is that your route into the company is often times some random developer within the company starts using your tool and then somebody else in the company starts using the tool and they're like, "Hey, we love this tool," and then over time the CIO or the CEO finds out about it and they say, "Cool! I guess we should buy this for you," and then so it's like you have a bottoms up sales processes. You have developers within that major enterprise, the bank or whatever saying, "We want this thing." The other model is the top-down sort of wine and dine the CIO model of sales. Have you started to think about this spectrum of ways to actually get Pachyderm adapted by a developers at enterprises?

[00:32:28] JD: Yeah. So we think about that spectrum constantly, because you're kind of forced to as you have new customer cases coming in. Thus far, the bottom up approach I think comes a lot more naturally to us because, again, like I said, I come from a background of open source. I'm a developer myself, so like when a random developer at a big company comes into our support channel, sends us an email or something like that and says, "I just deployed Pachyderm. I'm doing this with it. Here's what I'd like to be doing with it." That's a conversation that I completely understand that we know how to have.

When we have a CIO or someone like that who is poking around or when we're doing an outbound sale to a CIO or something like that, that's a lot more newer territory for us, for me personally. We've hired some people who this is their forte, and so they have those conversations now. Thus far, what I've found is that for a product like this, you can take that top-down approach, you can get the CIO totally sold and you might even close a contract based on that, but you're still not going to get anywhere like long-term and get the thing to actually stick if you don't get buy-in from the lower levels.

Even having a CIO say like to his engineers like, "Okay, you must write all of these stuff on Pachyderm. This stuff must all be migrated to Pachyderm." Even that sort of only gets you so far, which is sort of like was an eye-opening moment for me when I realized like, "Oh! Engineers like – There still even in big companies, where like they have all these structure around them, there's still kind of able to do their own thing. They still have a lot of autonomy." So if you can get them to actually want to build on Pachyderm, then to me is always the crucial part of the sale that doesn't happen even if you convince the CIO.

Now, the other thing that you can run into is that you can have an engineer who's totally invested in Pachyderm and has written everything in Pachyderm and they have like a production deployment that's using Pachyderm or something like that, and if you don't have the correct language to communicate the business value to the CIO, to whatever VPs are above this person, then you run into a huge problem trying to actually close this contract and get money out of them.

Thus far, we've taken the approach that we're much more okay with situations like that than the other way around. If we have a company that is using our product in production and we can't quite close the deal to actually get them to pay for support or get them to pay for the enterprise product, then that's fine right now. We're fine with having our open source product be our biggest competitor, and we often – Like every company, we're worried about churn and a lot of the churn that we do experience is people churning out of enterprise contracts and into the open source, which is way, way better than just churning-churning, right? Because someone who's using the open source product for a young product like ours, that's helpful no matter what. They're giving us feedback. They're filing bugs and they're sort of adding this legitimacy to it, right? Like those of the people who write blog posts about it. Those are the people that we can point to in references and things like that. So that's sort of the way that we approach that and it's totally a learning process to figure out how to actually close sales and stuff like that, something that wasn't in my background at all. My cofounder has that a lot more in his background.

[00:35:49] JM: Nice. Well, also, it's a nascent market. I mean, clearly this is something that people want, which is what must be exciting about it, but it's also very nascent as we've already kind of discussed.

[00:36:02] JD: It's very nascent, and I think one of the things that people don't realize about how nascent it is, is that probably more than 50% of the data projects that we see that people are interested in using Pachyderm for fail. Sometimes that has to do with Pachyderm bugs, but most of the time that just has to do with what they were trying to do with their data was too ambitious, not really workable with modern techniques.

One of the biggest stumbling blocks –

[00:36:29] JM: What's an example?

[00:36:30] JD: Of project that failed?

[00:36:32] JM: Yeah, because it was too ambitious.

[00:36:34] JD: So the most common case for this is somebody – we have tons of people that come into us and they say like, “We want to build sort of the platform that's going to power data science for our organization for the next five years.” Something like this, and they have this whole grand vision of like, “Okay, here's what the interface is going to look like. Here's where the data is going to be stored. Here's how people are going to be able to access it and everything like that,” and invariably they realize that what they're – Eventually they realize what they're doing is really an entire product unto itself, but they're trying to build it within a company targeted at the specific needs of the company and they just don't understand how big of an undertaking it is. This isn’t a lot a lot of ways what Pachyderm is, and this has been a company working on this for five years and that's all we’re focusing on. We don't have the constraints of another bigger company around us. We also don't have the resources of like a Fortune 500 company, but we still only satisfy a subset of the use cases. We’re not the system to rule them all.

The reality I think is that the system to rule them all really doesn't exist, like even if you look inside of Google, that's by far the most sophisticated type of like data company you’re going to find. All of these techniques, like MapReduce, distributed file systems, those all just come out of Google papers from 15 years ago. They still don't have the system to rule them all. They’re constantly coming up with new systems that are good at different things. They’ve got this like fragmented mess that works because – Well, they’ve got a bunch of really, really good engineers and they do have good infrastructure there.

But when you see your some Fortune 500 company that isn't really a tech company is sort of reinventing themselves as a tech company over the last 20 years and you're going to build this infrastructure to rule them all that Google has been unable to build, like that's just never going to happen. So a lot of times we have to do in conversations like that is like we have to sort of get people to think smaller and think one step at a time and say to them like, when somebody comes in and says they want to build something like that and they want to use Pachyderm to do it, we normally try to push them and say, “Okay, let's find one use case. Let's show that this can work on one use case and then let’s show that we can expand it to a handful of use cases, and then maybe we can start looking at all of them.” We can talk about those other use cases so that we make sure that we’re not just like cutting those off with some architecture decision that we make right now. But when the criteria for success is like the platform to rule them all, then that's almost always going to fail, in my experience.

[00:39:08] JM: Okay. There's a lot of really interesting stuff there. The data platform question is one that I have been fascinated by for the last couple of years, and you're suggesting that if the world follows the Google model. it will not be this big thick one tool to rule them all. It'll be some set of loosely coupled components. You've got Kubernetes over here. You've got BigQuery over here. You've got MapReduce over here. You've got to Apache Beam over here and you're doing all these different things and you're plugging them together however you want to, and maybe you're using something like Airflow or whatever – The Google – What is it? DataFlow thing?

[00:39:47] JD: DataFlow is –

[00:39:48] JM: Not DataFlow. Well, DataFlow and the Google context, that's like they're Beam thing, right? But they have some kind of scheduler that's like Airflow, right? Data sequentializer, whatever it it.

[00:39:59] JD: They have something like that, although I know – So one of the use cases that we're getting looked at for right now is basically an ETL engine on top of BigQuery, and that basically just be using Pachyderm pipelines. You wouldn't be storing any data in Pachyderm. You're just using containers to fire off, like container boots up and it sends a little SQL to the BigQuery and then it goes away, and the company that we're talking to about this, we told them like, "This will work. It's a container. You can run whatever you want in it. It's not really meant for this exactly, like this is the type of thing that I would expect BigQuery to have built into it."

So my understanding is that actually the data pipeline scheduler within Google's cloud is a bit of a missing piece right now. I think that like DataFlow will work for like that very specific like use case, but if you want just a general thing within there, I don't think that that actually does exist.

[00:40:50] JM: Okay. Well, anyway, the point I was trying to get to, because I want to get to data provenance and versioning eventually, that's a really interesting subject. But just a little bit more on this data platform side of things. Google is not necessarily like the rest of the world. I mean, maybe Google infrastructure for everyone is coming, but there are these – like a company like Dremio. Like I find Dremio really – Do you know Dremio? Have you looked at them at all?

[00:41:13] JD: I don’t think so.

[00:41:13] JM: It was originally based on the Google Dremel paper. It’s guys that came out at MapR. One of the early engineers was a guy who I think created Parquet. So you’ve got Parquet and Tomer from MapR who's done a ton of – He was at MapR for like five years. I mean, learned everything about data infrastructure. He’s like, “Okay. I understand how companies are managing things on the first generation cloud data platforms, Cloudera, MapR, Hortonworks,” like that's kind of where we’re evolving from. Well, in terms of like selling to the enterprise, like these companies have been working with MapR. They’ve been working with Cloudera. Now there’s a huge world of other data companies that are kind of coming and – I don't know, maybe from your point of view, disaggregating the big Cloudera installations, or just there’s plenty of companies who never even integrated with Cloudera and now they're becoming enterprise software companies, or you’ve got John Deere becoming a software company that sells factory equipment, for example, just to get my point.

So do have a sense, like will there be some thick data platform things that are useful? Because when you think of a company like Uber, Uber is in many ways like Google, but in some ways not like Google. Uber has a data platform. They have a fairly well-developed data platform that is kind of a one tool to rule them all, one data platform to rule them all. So is it the case where there are some enterprises where maybe you do want a one data platform to rule them all, and there're other enterprises where you want disaggregated sets of tools.

[00:42:50] JD: That's a very interesting question. I mean, I'm not specifically familiar with Uber’s data platform. I was pretty familiar with Airbnb’s, and we spoke to them. I would expect if we looked into the details of that, what you would find is that it is one platform in terms of like this is Uber platform, but then if you look at it, it's composed of a bunch of different sort of off-the-shelf patterns, off-the-shelf products or open source technologies, and there's a lot of different access patterns in there, and I think that that’s sort of the reason why things are never going to be quite as unified as you want, because you're going to have something like Dremel, which is a very, very fast access pattern for like columnar data, as I understand it, right? That is amazing for some use cases. For other use cases, it just doesn't apply at all. There's no way to apply that to video processing. There's no way to apply that for Uber’s self-driving car stuff, which is images and things like that, without a whole bunch of processing to turn that into some sort of a columnar format that people understand.

So I think that the kind of octopus nature of data infrastructure and stuff like that is almost inevitable because of the sort of fragmented nature of data itself. Really, the trick here is that we have this word data that is just one word, but can encompass so many different types of things. So that allows us to say like, “Well, a data platform. We have all of the things that are called data. Why don't we have the platform that can handle all of the things that are called data?” That just obfuscates some of the details of what data actually looks like and how, when you get into those details, everybody needs a slightly different access pattern on that.

That being said, I think there are aspects of this that you can make generic, and that sort of the things that we try to keep ourselves targeted on at Pachyderm, right? So all of these different data formats can sort of be represented as some like really vast swath of bites, right? So like these Parquet files are normally stored in HDFS is like a multi-terabyte file or something like that, right? When you can do that, if you can also store like a billion images as little files or something like that, like that can be a pretty generic interface. It still doesn't give you a generic access pattern, right?

So that's why we sort of try to get as close to a generic access pattern as possible by saying, “Well, our pattern is we give you a process in a container, or we give you multiple processes running in multiple containers, and so that's as open ended as we can make this so that people can run whatever code they want in there and then interface with other things,” and this still works really cleanly for some subset of use cases. Then for other use cases, you can do them, like you can spin up a Dremel cluster, or spin up a Spark cluster and then access it from within your container, but now you're kind of managing two clusters. There’s not really just Pachyderm or something like that and it gets a little bit less nice.

So I think that if there is a route to complete unification, it’s going to be through some sort of a generic scheduler like Kubernetes that can basically run all of these primitives on top of the compute in sort of the same type of environment. But to be honest, I don't I don't think that that's going to congeal within the next five years, maybe within the next 10. I mean, there's just too many sort platforms, like you can't run Red Shift in Kubernetes. There're a lot of people who – You could in theory if Amazon wanted to make, you're able to, but they don't, and there's a lot of special-purpose things on Google Cloud and things like that.

The different clouds actually I think are an interesting both source and sync of fragmentation like this. You sort of get – For new technologies, you get this fragmentation. So when containers came out, you had ECS, which is Elastic Container Scheduler. You had ACS, which was Azure Container Scheduler, and you had GKE, which was Google Kontainer Engine with a K instead of a C, because it was actually Kubernetes under the hood, right?

Now, you have EKS, which I believe also stands for Elastic Kontainer Scheduler, but it's a K to indicate that this is the Kubernetes version of the container scheduler, and you have AKS, which is the Azure equivalent. So you saw like a full cycle over about three years of containers come out, container scheduler fragment, Kubernetes wins, container schedulers defragment, but they're still not totally defragmented.

Yeah, like they're still just a bunch of different ones and there's still like EKS, AKS, GKE. We have deployments on all of them. We've work with customers to get those deployments working on all of them. It's always slightly a little bit different. It's close enough that the dream does work. We do have a product that we basically maintain for all of these platforms, and probably 99% of the code that gets written you don't have to care about which platform it's running on.

So it's just that little 1% that you have to deal with. But this is container schedulers, which were like the hottest, most important field, where like you know inside of every single one of these clouds they had boss saying, whoever was in charge, like, "Containers, guys. Containers need to work with. What's our story for containers?" So that's how they were able to ship the container scheduler, and then the new container scheduler when they realized the old container scheduler wasn't what the market wanted. But for data, you don't have anywhere near that level of pressure, right? People know that data is important. People know that machine learning is important, but there isn't as clear of a waypoint for people to say like, "Oh, yes. Kubernetes. This is the API the people want. If we support this API, like people will be able to use off-the-shelf products on our cloud. Things will be good." So there's a lot more fragmentation. There's a lot more of companies saying like, “Oh! Well, maybe we want to make something like SageMaker. SageMaker is sort of a very automated. It applies a bunch of different machine learning algorithms as something. It tells you which one is best. It's kind of like a machine learning consultant in the cloud is may be a way to phrase it,” that that's one viable hypothesis for what people want out of this market. So as long as there’s enough sort of viable hypotheses, you’re going to see this fragmentation at the cloud level.

We sort of have a thesis on this ourselves as I said of like you want to sort of have a data aware scheduler that gives people a very, very generic way to specify what their data operations actually are. For us, the next step of this is sort of bringing it on to the cloud. So with the series A that we just raised over the next year, what we’re building out is a site called PackHub, which is basically if you think of Pachyderm as like git for data, then PackHub is like GitHub for data.

So it's sort of an online hub that will connect both a large Pachyderm instance that we’re running and individual Pachyderm instances that other people are running in their clouds and allow you to sort of push and pull pipelines and publish pipelines that you’re using to do data science, push and pull data and see results and things like that and connect sort of the users on top of that so you can come in to your office and you’ve been working on PackHub for the last five years while you're in school and they’ll say, “Okay, we’re giving you access to our PackHub org,” and now you can see all the pipelines that they have running and all of the data and everything like that, and it's a unified interface for this.

But it's a long road toward people being able to do all the different things that they want to do with data on PackHub.

[00:50:15] JM: Why hasn't that been built before?

[00:50:17] JD: There have been lots and lots of companies that were building GitHub for data science, or GitHub for data. So it has been built before in that sense. You can go and look at those things and none of them have really taken off, because in my opinion they didn't really deliver on the promise of what that could be.

So this was one of the very, very early thesis for our company of we always wanted to build something like that, and we consider just diving in straight ahead and building like, “All right, let's build a website. Let’s build this portal. Let’s build something like that,” and what we realized is like, “Okay, there's a lot of other people doing this and they don't seem to be succeeding that well. Why do we think that is?”

The reason that we came up with is that you can't build GitHub until git exists, right? Git had to be written and git is just sort of this like miracle nice thing that Linus Torvalds decided to write for us, and it's this amazing version control –

[00:51:12] JM: And it accidentally became a social network.

[00:51:15] JD: Right, and it accidentally became a social network. It accidentally became the de facto way that people do open source development and a lot of other kinds of development. Of course, even when they're not using GitHub, I think the main thing they're using is GitLaB, which is just another take on git as this. I don't think there's much. I still hear about mercurial and stuff like that once in a while. I haven't heard about Darcs in a while, which was the Haskell version control system.

[00:51:39] JM: I don’t know what that is.

[00:51:41] JD: Darcs? I’m a big Haskell nerd, or at least I was. I can't really legitimately say I’m anymore. So Darcs is this kind of cool version control system written in Haskell that has this really fatal flaw where every once in a while it tries to solve an NP complete problem on your commit graph. So that just takes like days. There're some issues there, but we sort of looked at all of these other like GitHub for data attempts and realized like, “These guys are kind of building UIs on top of the same existing tools that we feel aren’t really solving the problem.” We don’t see anybody who's really willing to like grapple with the hard problem of what does the underlying infrastructure that the UI is depending on have to actually do to facilitate like data collaboration between eventually the world. It's not every person in the world is not going to be using this product, because there’s a lot of people who just have no interest in writing code that processes data. But on the data processing world, that’s still like billions of people.

So you need really, really high-power underlying infrastructure that can version control large datasets, that can do distributed processing on that. Incremental processing becomes like hugely important on that, because like you're going to have all of these different workloads of like the guy who's updating his code like every five minutes as he processes it, versus the guy who has written his code once and it’s correct and he deployed it, but new data is coming in every five seconds and that needs to be processed.

So you need very, very sophisticated underlying distributed systems primitives to make this system actually work, right? I mean, that's the reason that people use GitHub is because – Did you ever use like SVN or sort of the predecessor systems?

[00:53:27] JM: Oh, yeah. I've entered SVN into the command line before I was really confused.

[00:53:31] JD: Right. Yeah. To be fair, the feature of git is certainly not that it doesn't confuse you, but the feature of git is that it's a workhorse. Like when you tell it like pull this stuff, it might take a while, but it's going to pull all of it, like it's going to do all – It seldom –I've never seen git break on me for reasons that I didn't feel like were my fault. It’s its fault for confusing me. The user experience on that is what to expect when Linus Torvalds writes a system for Linus Torvalds to use?

But it's never just flat out broken on me, and that's a lot of the problem that I saw plugging in data infrastructure just at a single company level at Airbnb. So if a single company can’t maintain data infrastructure for just the people within their company to use, how is a company going to maintain data infrastructure for the broader public to use when they can do whatever they want on it and all sorts of crazy things that happen and the scale is that much larger?

We felt like we needed – And this was what I was excited about and founding this company was there's a lot of really interesting hard systems problems to be solved here, and those are the problems that I have a background in. Those are the things that I really like to solve. So I wanted to pick a company to found that where focusing on those problems actually made sense, because I knew there was a huge risk if I founded a company were actually the correct answer and their correct product was like, “This should be a website first and then we'll figure out the back-ended infrastructure later,” then there was a good chance that I was just going to focus on the infrastructure, because that's what I'm interested in and I completely do a terrible job with the company. So I tried to pick a place where my instincts were going to leave me in approximately the right direction.

[00:55:19] JM: And the giant market. You get the giant market. You got the self-aware playing to your flaws as well as your strengths.

Okay, we've been talking in like the meta-realm for basically 52 minutes and we have like 10 minutes left, which is great, like people love the meta-realm by the way, and we definitely have to have you – I want to have you back as a guest like much, much sooner, because this is like –

[00:55:40] JD: I love to.

[00:55:40] JM: Yeah, it'd be great. But let's talk a little bit about data providence, auditability. This is not stuff we've done a show on. It's definitely we should do a show on. Frankly, it sounds really intimidating. Like if I'm a bank and I have a machine learning model that's issuing alone and I don't know how that model has been trained. I don't know exactly what datasets it's been trained on, and then the model is getting versioned over time. There're different versions that are being served to different people. I might have A-B testing going on. I've got this insane tree of past decisions, and then if I get served with a lawsuit from somebody who says, "I got discriminated against on November 15th at 2 PM when I applied for a loan," and I'm like, "Oh my God! I have no way to know which version of the model was deployed then. I don't know which one was A-B testing," then – I don't know. Do I have to get their IP address and then like map that IP address to it?

So you get all these issues around auditability that are becoming more and more relevant, and in like the loan, the loan example is almost a trivial example compared to what we're going to be dealing with in the next couple of decades when you get into like healthcare and like real-time things that are going to like hurt people, and like hurt people in real-time, kill people in real-time, when you're going to be like, "What line of code was to blame for that?" and you're like, "Oh! It was the machine learning model that made a judgment," and we have no idea why it made that judgment.

So with all that said, I don’t know, take me into the shallow end of the rabbit hole that is Joe Doline’'s opinion on machine learning auditability.

[00:57:30] JD: Yeah. So obviously this is something that we deal with every day and is a big part of our value proposition. So I, by thoughts on it, have sort of been developed by that position that we’re in and we try not to spin too much FUD for companies. We always try to give them as realistic an impression as we can of what do you actually need to do? What is going to be expected of you for this stuff?

The tricky thing though is that since the GDPR passed, we really don't know. That whole scenario that you described of like someone comes in and says like, “Show me what's happened on this [inaudible 00:58:08],” or something like that. That is all laid out in the law as something that can happen. We haven't seen cases go through the courts, right? Of course, you're talking about a bunch of different courts, like the GDPR passed for the EU, but when this gets tried, I'm pretty sure it's going to be tried in an English court, in a French court, in a Portuguese court, things like that.

So you're going to see completely different results on all of these stuff. So the thing that we always try to tell people is like, “Look, you need a system that just tracks everything, and you need a system where the tracking isn't something that your data scientists do as part of their code, because these sort of existing systems like this is something like Apache NiFI. That's really the only other one that we know of that's like their whole thing is providence, and that's like an Apache project. I’m not even sure if there's a company behind it. There will be eventually. Every Apache project eventually becomes a company.

[00:59:04] JM: Not ZooKeeper.

[00:59:05] JD: That's true, not ZooKeeper. Every new Apache project that wasn't founded before Cloudera and Hortonworks I think eventually becomes a company. The ones that were around before just sort of got rolled up in Cloudera and Hortonworks.

[00:59:18] JM: Right. Well, ZooKeeper powers all the companies.

[00:59:20] JD: Right.

[00:59:20] JM: Which is hilarious.

[00:59:21] JD: I probably mentioned this on the last podcast how like that is really like the perfect example of the biggest systematic problem with the Hadoop ecosystem, that ZooKeeper powers all of these and there's no company behind ZooKeeper. So when we would have outages at Airbnb, so many of them would be caused by ZooKeeper and we'd be like, "Can somebody fix this bug with ZooKeeper?" It's like, "Well, why would we do that? That just makes their product better too. ZooKeeper isn't our thing."

[00:59:46] JM: Unfortunately for you, arguably. Now, that is etcd's responsibility, which is CoreOS's responsibility.

[00:59:55] JD: Which is IBM's responsibility.

[00:59:57] JM: Which is IBM's responsibility, who you just dissed.

[01:00:00] JD: Yeah. There is that thing. I mean, I think the etcd – The thing that we have going for us on this, though, is that etcd also powers Kubernetes. So if etcd really has fatal flaws in it that are preventing it from servicing like big workloads, it's probably going to bite the biggest Kubernetes installations before it bites the biggest Pachyderm installations, and I think Google is going to have some motivation to get that fixed.

My understanding is that etcd is kind of now in its own foundation spun off. Like if you go to the GitHub project, it's not under CoreOS, Red Hat or IBM anymore. It's its own thing. Anyways, this is a risk. This is a real risk.

Let's get back to provenance. So like I said, we try not to sort of spin too much FUD for people, but the thing that we pitch them on about Pachyderm is that this is a system where provenance isn't something you do. It's something that just happens. So when you're naturally using Pachyderm, there's no way to turn off the provenance features. There's no way to forget to record the provenance of this piece of data. When you process a piece of data in Pachyderm, it always has all the provenance attached to it of every single line of code that ran for this and every single piece of data that went into it. So if you want to, you can just sort of click a button and say like, “Let's re-create this process,” and you can drop in there and see the logs coming out and actually debug it and things like that and you can play with the results as they come out and see if they're the same.

So this is sort of the system for the people who were paranoid about this stuff. We want them to feel safe when they use this that there's really no way that you can use Pachyderm and not have complete auditability if the data comes out as long as things stay within Pachyderm. Where this starts to break down is when people's sort of start to mix Pachyderm with other stuff, which of course there's always a reason to do, because we don't support all paradigms yet. We’re trying to sort of support as many of them, but sometimes you want to have the distributed model of Spark or something like that. So people will check data out of Pachyderm and then process it in Spark and then sort of check it back into Pachyderm, and then we can say, “Look, this works totally fine. You're just taking out a tiny bit of reliability and that we can't say 100% how this was processed. We can't re-create this processing if you did it manually yourself.”

Similarly, sometimes people will have like a large swath of data that’s sitting in some distributed file system that they have, like [inaudible 01:02:22] or something like that, and they like – They’re like, “Well, we don't want to ingress all of this into Pachyderm before we can start processing it at all. So could we have our storage and our pipelines basically have a pointer to it?” and they just say like, “Okay, this data is over here in [inaudible 01:02:36],” and when we want to process it, we know how to download it from [inaudible 01:02:38] and then we write it back out to [inaudible 01:02:40] or maybe we write it into Pachyderm, but we sort of record this pointer and we always tell people that that's totally fine, that's going to work. The only risk that you're taking on is that we can't guarantee that the data hasn't changed in [inaudible 01:02:52]. It’s just a dumb pointer where we know this data is here somewhere and we’ll tell you about it, and if it suddenly changes, then when you go to audit it, it’ll have changed.

[SPONSOR MESSAGE]

[01:03:08] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[01:04:32] JM: This is going to be an amazing business, because you're going to be like one of the categories like distributed tracing or log management where every company is going to need this and there’s going to be reams and reams of data in this system.

[01:04:49] JD: That's how we feel, and it's designed to be sort of snowbally, if you like. Meaning that like it sort of rolls down the hill and gets bigger and bigger and bigger as you put –

[01:04:59] JM: You mean, the company, the product?

[01:05:01] JD: Specifically the product, like the file system and the data, the data infrastructure and stuff like that, because like you put some data into it and then you process some stuff and then you have traceability with that and then you want it to go somewhere else. It’s like, “Well, if we don't put it within the same system, we won't have traceability.” So we want people to be able to use this is as sort of the base layer for all of their data processing, storage and data science so that they always have just this clean provenance line from ingress to egress of where it went out into the outside world. Yeah, we think it's going to be an amazing business. That's why we’re doing it.

[01:05:34] JM: Amazing. Okay. We're almost out of time. So now that I've shine my praises on the company, and you just raised the $10 million series A, which is amazing. Series A after like four years of running the company or five years? I mean, what I like is I get a sense that there was a lot of persistence that was required to get to this point and a lot of long-term thinking, because now I think you're probably in a point where you're not like out of the chasm of despair completely in terms like product building, but like you're certainly beyond a certain chasm that you were in before.

Do you have any reflections on like – Because most companies don't – I mean, in the Valley over the last four or five years, it's not a four-year timeline to a series A. It's like you work for a year and a half and then you're acquired or you die or you raise a series A. Yeah.

[01:06:33] JD: Yeah, that is like the I think more standard timeline. I think that gets a little bit skewed by how things get reported and things like that. We raised our series A as part of YC's like series A program, which is basically like traditional YC, except at the end you raise a series A instead of C round, and one of the older companies in there, but not the oldest, there were a couple of companies that had been around longer than us and I think wound up racing series A's as well. But you're absolutely right that it did require a lot of persistence and we are on the longer end, and I think that part of that is that people often pick business models or at least I think the people do and I felt myself being at risk of this and avoided it of picking a business model that is designed to be able to grow really, really fast in the short term, but then when you get to like the series B, like that type of range, it just totally peters out because there isn't actually like a long-term viable business there.

So I think a great example of this that I've seen this kind of in our space is logging companies, basically companies that will like to give you an endpoint to send your logs to and be like searchable for it. I keep seeing companies getting founded around that premise, and I think that a lot of the reason is that you log – Storing logs is a big problem. It's often a pretty expensive problem. It's annoying to run, like the ELK stack and stuff like that. So what I would assume is that you can always – Like when you get a system like this and you're going to sort of like subsidize people's log storage and search and stuff like that, you can get enough people to come on to that system that you can show the metrics that you need to raise a series A under the promise of like, "Well, we're going to have everybody on this log system and stuff like that." I haven't seen any really like logging companies like quite work out like that. I mean, I think like Splunk is the most successful one and they're like – What? 10-years-old now?

[01:08:33] JM: Well, I mean the advantage on the logging side of things is logging is a deep and subjective problem to solve, and there's so much depth to it that it might be a category unto itself. Like we look at the cloud provider market today, like it's a giant market, but obviously in 10 years it's going to look way, way, way bigger. Log management might be the same things. So the companies that are getting started today, they might just be like, "I've have been doing log management. I have tried Splunk. I've tried Datadog. I've tried New Relic, and like either these things are doing too much, or they're doing too little, or they're not doing distributed tracing, or they not doing like machine learning insights, and it's like, "I have a new take on logging, and I think it can get me this series B, and it also happens to be a business that can give me some quicker dopamine hits." I don't know.

[01:09:20] JD: Yeah, and I think the, one, I won't comment on this like – I won't commit to anything until I really see how this market shapes up. There're still all these logging companies and like maybe one of them is going to turn out to be really, really successful. But when I look at a business like that, to me, I just can't quite make the argument of like what is the long-term like really, really valuable thing that we're making that nobody else is going to be able to make, right?

So I wanted to focus on a business where I felt like this is something that needs to be made and people are trying to make it and they're failing at it, and not in terms of like the machine learning insights aren't quite good enough, or like it doesn't work as well for this use case of just like it just keeps breaking and people aren't accomplishing what they need from it on a very, very basic level.

So I wanted to focus on something that I felt like could be a very, very long-term like successful, really, open source project was the first thing I wanted to focus on. When I sort of was thinking to myself like, "Okay, if I start working on like the best logging solution in the world right now, can I imagine that like we're going to have this successful open source project 10 years from now that's like the logging thing that everybody uses?" Like probably not. I think like the ELK stack is like probably going to incrementally improve, and like that's going to be the way that people do it most part and you're going to get additions on top of that, things like that. Versus the data platform that everybody can use to run and process their data and have provenance and stuff like that that. That seemed like something that didn't really exist at all.

So I wanted to focus on seeing if I could actually bring something like that to market and have that be something that could really be long-term successful and stand the test of time. A lot of the reason that the series A took so long to raise and putting the company in a position where he could do it took so long is that there's a ton of technical work that needed to be done there. So we needed to build out all of these actual infrastructure to have when it's still sort of in certainly not a prototype state, but it's not in its final state either.

It’s in a state where it’s useful to people for some subset of the use cases we want to ultimately target, but it also took time for the market to develop around it for people to realize that when we were first going out to raise our seed round, a lot of the people who said no to us said no because they’re like, “I just don't believe that anything can ever beat Hadoop. I think that Hadoop is like the end-all be-all,” and that was an objection as recently as like three years ago, and it felt like all of a sudden things just switched. People were like, “Okay. Hadoop is old news. We’re interested in something new now. Like we feel like that ecosystem is vulnerable.”

I think actually the Hortonworks Claudera merger might've helped a lot with that, at least in terms of like the VC world, because it sort of showed like, “Okay, we are very much in the consolidation phase of this network or of this market. We’re not necessarily in the like huge high, like explosive growth phase.” They're still going to grow and things, but when the two sort of major competitors decide to call it quits and stop like competing with each other to win the market and just like, you know, mend bridges and try to get as much value out of it as possible, like that’s signal of something.

So it took some time for the market to develop and it also just took some time for us to find the right sort of investor for us to work with. We weren’t in a huge hurry to just get money in the door for getting money in the door. We basically, for the last two years before we raised a series A, we had a small but effective development team. We had a core set of users that we were working with and we were getting enough money from them to mostly fund the business. We got a little bit of help from our seed investors sort of kicking in some more money midway through that period. But most of it was just climbing the mountain, doing the hard work with our customers and keeping the company afloat during that, because –I felt like that sort of push the company to get good number of things that we wouldn't have gotten good at otherwise, I think that if we just been able to raise a series A after like two years and been on this sort of like highflying trajectory, we would've learned what it actually takes to close a sale and how much it hurts to have someone churn out things like that, because we – We're just like, "All right, whatever. We'll just raise the next round. Let's hire some people. Let's do some stuff like that." So startups take different paths. All we know is the one that we're on, but so far I'm pretty happy with it.

[01:13:46] JM: I think you're in a great spot. I wish we could talk more, but I got to get going, because there's another interview. We'll do this again soon. Joe, it's been really fun talking to you.

[01:13:54] JD: You too, man.

[01:13:55] JM: Okay.

[01:13:55] JD: Thanks a lot for having me, Jeff.

[END OF INTERVIEW]

[] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[END]