**EPISODE 1009**

[INTRODUCTION]

**[00:00:00] JM:** In a modern data platform distributed streaming systems are used to read data coming off of an application in real-time. There are a wide variety of streaming systems, including Kafka Streams Apache Samsz, Apache Flink, Spark Streaming and more.

When Eric Anderson joined the show back in 2016, he was working at Google on Google Cloud DataFlow, which is a managed service for handling streaming data. Today, Eric works as an investor at Scale Venture Partners. In his current job, he analyzes companies built around data infrastructure, developer tooling and other enterprise engineering domains.

Eric also hosts the podcast Contributor, which explores open source maintainers and the stories of their projects. Eric's podcast has featured the creators of projects such as Envoy, Alluxio, and Chef. In today's episode, Eric returns to discuss data infrastructure investing and the evolving world of open source.

[SPONSOR MESSAGE]

**[00:01:08] JM:** Logi Analytics believes that any product manager should be successful. Every developer should be proud of their creation and they believe that every application should provide users with value. Applications are the face of your business today and it’s how people interact with you. Logi can help you provide your users the best experience possible. Logi’s embedded analytics platform makes it possible to create, update and brand your analytics so that they seamlessly integrate with your application.

Visit logianalytics.com/sedaily and see what’s possible with Logi today. You can use Logi to create dashboards that fit into your application and give your users the analytics that they’re looking for. Go to logianalytics.com/sedaily.

[INTERVIEW]

**[00:02:06] JM:** Eric Anderson, welcome back to Software Engineering Daily.

**[00:02:09] EA:** Oh! That's right. Yes.

**[00:02:11] JM:** Yes. You were on a while ago. That was back in 2016 when you were at Google Cloud. We talked about Google Cloud DataFlow, and you were working on the Google Cloud team for a while, for about four years. You were working on the data engineering products. How has the data engineering ecosystem changed since those days?

**[00:02:36] EA:** I suppose when I was just starting, Hadoop still a big deal. There was the transition the Spark where everyone was kind of rebranding the ecosystem Spark, but that's more or less the same ecosystem. Then the unexpected rise of data warehouse was a big deal, where I think it took all the vacuum out of the room. I think that basically we thought we'd had to build this complex systems in order to store massive amounts of data and then we had these programming languages and processing systems to get out of them, get data out of them. With BigQuery, Red Shift, and Snowflake, I think a lot of people were like, "Oh! We can just throw in the database just like before and create with SQL." That gets us 90% of what we wanted.

I think the rise of data warehouse, it's been a big deal. The rise of Python. So when I started the whole data engineering world revolved around the JVM, and Hadoop, then Spark, then Kafka. These were all big Java monoliths. Meanwhile, data science was kind of doing its own thing over NumPy, SymPy land, and that's evolved in machine learning and that's become the primary use case for data engineering. So that now Dask another kind of Python projects are our center stage, which I find kind of exciting to have a kind of a programming language driving new tech shift.

**[00:03:59] JM:** On the data warehousing side, you have Spark, the Spark people, the Databricks people who have built Delta, which I think combines data lake with their processing system. Then you have the Kafka people who have stream processing systems, KSQL, built on top of Kafka. Then you have the Snowflake system, which is as I understand unified data lake and data warehouse. You have all these different paradigms of how people do data warehousing. Is there consensus for what is desired the most out of a data warehousing system?

**[00:04:46] EA:** Those systems you mentioned started very differently and then they kind of converged this SQL queries on a big bunch of data to some degree. I think what happens, you have really talented engineering teams with very specific requirements to accomplish, whether that's ingest a stream of data, or to do reoccurring processing on a bunch of data. They build these amazing systems, highly available, very fast and efficient. Then you have the whole data science community just wants to query data and they want to then consume these highly available, real efficient systems. So those systems evolve from being very complex to operate and rigorous in the code the write to use them to being eventually get a SQL layer on top and you get data layer on bottom and it starts to look all – They all kind of start looking like a data warehouse, because that's how a bunch of the enterprise wants to consume things. I find that interesting. There's a semblance of convergence and that the enterprise data science community keeps pulling us towards how to SQL database for data engineering and data science.

**[00:06:00] JM:** Are there particular gaps in the data infrastructure tooling that you see?

**[00:06:06] EA:** Certainly. There're still operational gaps. When I look at opportunities for investment, for example, now, I kind of look at this lens of like where things missing in data engineering community. Kafka is super popular. Operationalizing a Kafka cluster is a momentous fete. People pay companies in tens of millions of dollars to keep Kafka up and running for them.

I think there's an opportunity in giving the enterprise or just the world a kind of highly efficient and easy to operate stream processing service. There is maybe one. Another is I feel like orchestration, if you want to call that, is still troublesome, like Airflow I think was very popular tool. Still is.

**[00:06:58] JM:** Yeah.

**[00:06:59] EA:** But I feel like there is still a little dissatisfaction in the world of like – I use one system to kind of describe my job. Another system to kind of tell it when to start and stop, and

there's some dissonance between that idea that like I just want this thing to tell another what to do. Like retries get tricky, which is kind of the fundamental thing you're trying to solve for.

I think that kind of an all-in-one orchestration and processing tool could be interesting. Then maybe the third area would be the world of data science still feel separated from data engineering. We've been talking about this for a while, like putting models into production is harder than it should be. That feels like still a ripe berry for an opportunity.

**[00:07:40] JM:** Those are all great points. The productionizing of a Kafka-like system, why doesn't Kinesis or Google Cloud Pub/Sub, or Confluent's Kafka offerings, why don't those do the trick?

**[00:07:58] EA:** Put differently, why are people running Kafka open source if it's so hard given these cloud provider offerings? There're still some concerns people have with running. These are super important systems for people. Turning them on to a cloud provider is a concern. I think pricing ends up being a concern. I know in our experience that as soon as you throw a lot of volume at one of these from like when I was at Google days, as soon as a customer puts a lot of volume into this, they start coming back and asking about pricing.

**[00:08:29] JM:** So it adds up.

**[00:08:29] EA:** It adds up. Yeah. Maybe that's where I think efficiency is one of these hallmarks of this opportunity. If you can continue to press the performance needle, there's an opportunity I think to steal market share, because there's a lot of people, like I said, paying a lot of money either to Confluent to keep it up or to the cloud providers for like a per message basis at volume.

**[00:08:54] JM:** And the Airflow point, the point about orchestration, all in one kind of scheduling, or I guess doing – I think the Airflow, the thing that Airflow solved was you have a series of things that you might need to do to get an end-to-end machine learning or data engineering workflow where like maybe stage one you do some kind of ETL from a data lake into a data warehouse. Stage two, you do some work in that data warehouse. Maybe you extract some clickstream analytics. Then stage three, you queue up a machine learning job in TensorFlow. So

if you wanted to wire those three operations together, Airflow accomplishes those series of steps. But there're some pain points in the integration process along those series of steps. Is that what you're saying with the Airflow issue?

**[00:09:55] EA:** Yeah. So we have these awesome, powerful engines, Spark, what have you, that do processing and scale efficiently. Everyone wants to use those. But they were designed to handle these kind of nuance-y little scenarios like, "Hey, I want to get the answer to this and go shove it over here and do this other thing. We say go find an orchestrator, Airflow.

But then Airflow – Mostly what these former systems were really good at was retries, was like ensuring that the job got done or didn't get done idempotency basically that like if you're going to – Because that that's the problem with a distributed system, is that you're going to run a task. Some of us is going to complete. Some of us is not going to complete. We're going to retry the parts that don't complete, and so that when the job is done, it's done, regardless of like node failure or other concerns.

Orchestrators aren't going to worry of that. That's a very tricky thing for them to do. They're going to leave that to the other systems. So the assumption that Airflow has to rely on is that everything it executes is idempotent. It's re-triable and it won't finish unless it succeeds, and maintain that guarantee across a series of steps is kind of annoying as a developer. You're like, "Well, I just want that, the results of this thing, to go into this database and then do this other step. If that other step doesn't work, I want to go back to the step two steps ago and create a new value so that that step then works."

That then – I guess, because these three steps have side effects and are not idempotent, I need to make them just kind of one Airflow step. Then like, "Well, now I'm not really using Airflow much, because my multistep task is really just one Airflow task. So now I'm back into Spark and I'm doing complex kind of MapReduce-y-like code in order to accomplish my orchestration job."

Yeah, I think a system that kind of think both ways and maybe there's opportunities with Dask now becoming as popular, emerging in popularity, to –

**[00:12:00] JM:** What is that? Dask?

**[00:12:02] EA:** Dask. Yeah.

**[00:12:02] JM:** What is that?

**[00:12:03] EA:** Dask is a big data processing service like Spark, but it's Python-centric. You have PySpark, which is like a Python API on Spark. I want to attribute to the Anaconda team and the – Which I want to say are behind NumPy and others develop this open source project called Dask, D-A-S-K, which is a parallelizable job execution, a service, but it's all Python from the ground-up, I believe. If not, just like mostly python API. Perhaps there some more efficient stuff underneath. But, yeah, it's becoming more popular. I think it's becoming more popular because it's not Java and it's not tied to all the tooling that was kind of held up the Hadoop Spark world.

**[00:12:50] JM:** But if you're using PySpark and you're just interacting with the Python APIs, why is any of that exposed to you? Why is any of the Java ecosystem actually exposed to you? Isn't all abstracted away from you?

**[00:13:03] EA:** It is. I think from like – As a job developer, I think PySpark, it's mostly Python. I don't know if like in debugging, if you ever end up with like low-level errors that you're like, "Oh! What does this actually mean?

**[00:13:18] JM:** Right. JVM stack trace. Oh no!

**[00:13:20] EA:** Yes. Yeah, that may happen in PySpark land. Maybe not. But I think, certainly, as like the DevOps team that's managing the Spark cluster, they're still managing a Java monolith. If you're a data science team who wants to execute parallelizable jobs, you can either hire someone to manage Spark for you and execute your PySpark, or you can just run to Dask yourself, because you may feel more comfortable managing a Python-centric system.

**[00:13:47] JM:** You spent a little time working AWS before going to Google Cloud.

**[00:13:51] EA:** Yeah.

**[00:13:52] JM:** How does the Google Cloud culture compare to AWS?

**[00:13:55] EA:** Oh, I could talk for a while about this.

**[00:13:58] JM:** Please do.

**[00:14:00] EA:** I have like a whole lot of respect for both cultures and they excel at different things, which is why they're both trillion dollar companies doing amazing things for us. One notable difference is that there's a ton of autonomy at AWS. They have separate teams that are all kind of competing on their own to accomplish whatever they're doing, which can be a kind of a complicated and environment to work in where you're like, "Hey, they're doing kind of what we're doing. We should go talk to them." But you might be like, "No. It's not. Let's just beat them. Let's out execute. Let's just deliver on our promise. They have their own kind of charter."

I think you see that in how the product growth portfolio just proliferates. I mean, there're just products that are kind of like other products. Nothing gets retired. It's just more and more. This is a world of like super capable, highly autonomous teams just charging at the world and shipping things.

On the other hand, Google has always been like a consensus-driven place. If you see another team shipping something similar to you, you go talk to them and you kind of have like a Kumbaya moment where you talk about, "What are you trying to accomplish? What are we trying to accomplish? There are ways we can work together," and then you kind of share that up the chain of command and there are several kinds of off society meetings where people come together and agree on things, which addresses the former concern. Maybe it's a pleasant place to work because you feel like we're all working together. But things don't quite ship as quickly or as easily.

I mean, if you want to ship something, you have to not only decide what customers want but you have to decide what the other teams are trying to do when coordinate in complementary ways and make sure everyone's on board with all this not only from like a cross organizational perspective, but I think engineers are more empowered at Google. So you have to get

everybody on your engineering team onboard. I think shipping at Google is – And maybe this is a product management point of view, but it's a whole bunch of consensus building.

I think that's – Like I said, it excels in shipping – You look at the GCP product portfolio. There aren't as many products, but I think there is a lot of harmony. There's a lot of thought and eventually you arrive at a really awesome product. You arrive a little slower. I think people fall at Google at not moving quick enough relative to AWS. That's probably the biggest –

**[00:16:24] JM:** It's more opinionated, right? Google Cloud is more opinionated.

**[00:16:27] EA:** It is more opinionated. Yes.

**[00:16:29] JM:** Is that all – The cloud that Google is building, is it asymptoting towards a Google infrastructure for everyone type of thing or does Google sort of acknowledge that not everyone is Google. Not everyone will want Google –

**[00:16:47] EA:** Sorry. I'm not sure how to help.

**[00:16:49] JM:** Not everyone will Google infrastructure for everyone. Does the market want the same things that the internal Google engineers want?

**[00:17:01] EA:** No, and certainly not the exact same and, and that's been – What was the mission to ship Google for everyone? Initially, yeah, I think so. I think the team took the kind of Bezos mentality of innovating on behalf of the user, like serve them what they really want. Not necessarily what they're asking for.

**[00:17:21] JM:** The AWS team.

**[00:17:22] EA:** Yeah. Sorry. I'm actually talking about Google.

**[00:17:24] JM:** Oh, okay.

**[00:17:25] EA:** Originally, they took that a little too far. They served the world of PaaS. That was kind of the app engine, like Azure, was kind of what we thought the world needed. That turns out that the market for infrastructure and just servers was so much bigger and the world was so much more ready for it.

Anyway, that's just kind of one well-documented point in history. I shouldn't spend time on that. But I think Google continued to ship next gen products in a world that just wanted like awesome, current gen.

**[00:17:59] JM:** Today's product.

**[00:18:00] EA:** Yeah. I think with leadership [inaudible 00:18:04] trial and error, increasingly Google's like now doing interesting things where they're like, "Can we adjust our – Take what we know about managing awesome opinionated stuff in a certain way, and then can we also take what we know about what people want today? We deliver something that's like awesome, current gen tech."

I feel like Kubernetes was along that line. Pub/Sub, for example, I think is mostly just an API on Google's internal pub/sub. But we didn't do that with Kubernetes. Kubernetes is a new ground-up project inspired by Borg, but we didn't ship Borg. It's a new thing. I think that's the –I would imagine, or that feels like the kind of ammo going forward, is let's take what we know about what we've learned from Google building Google tech, opinionated amazing stuff, and like let's use that as an inspiration for incremental products that the market is ready for, which feels like a good approach.

**[00:19:11] JM:** Do you know in what ways Kubernetes diverged from the Borg plan?

**[00:19:22] EA:** No. I mean, I'm not very familiar with how Borg operates. I'm only somewhat familiar with how Kubernetes operates, to be honest. But I do know that to make an engineer productive on shipping stuff on the Borg took like months. You think Kubernetes YAML is bad. Defining a job from Borg was like this complicated sea of levers.

As you can imagine, any kind of internal project tends to be. It's like we don't need a lot of polish on this. We don't need to constrain what users can do, because they can do whatever they want. It's internal. So just like give them the world the can – The control panel, all the widgets you can imagine. There were like wikis upon wikis about like how to get something through build and on to Borg. Very positive innovation in terms of like shipping a more opinionated and kind of constrained artifact in Kubernetes. There's probably a bunch of like maybe low-level performance significant things that are different, and I'm not as familiar with those.

[SPONSOR MESSAGE]

**[00:20:32] JM:** Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

**[00:22:21] JM:** The Google DNA I think is shaped by the fact that it has a money printing machine in the advertising business. Amazon's core business had much lower margins, and overtime they have built what seems like their own money printing machine in the cloud business. But do you think those first core businesses, the fact that Google had a money printing machine, Amazon had to grind it out in the retail business. Did that affect how those companies have built their respective cloud provider businesses?

**[00:23:04] EA:** Gosh! It's like your team. Yeah. No, this is like something I think a lot about, and I actually really like that question and I agree with the way you phrased it. Things like Google had Google translate for a long time. They had voice recognition for a long time. They had pretty sophisticated tools and they surface them to their users. I imagine they felt like that drove traffic to Google and it fed the machine, the money printing machine. But when it came time to like, "Do we sell those directly so that someone can make products that compete with our – That could attract attention? Maybe even a way from our money printing machine?" I think that line of argument always slowed Google down, or it eliminated Google's desire to pursue a market like AWS. It was like, "We can ship app engine, because that's kind of like the – Yeah, kind of weird, wonky thing. But do we want to ship Google Brain or even DataFlow, I think?" These were all seen as like, for a while, Google published papers as supposed to open source projects in part because it was like, "We want our PhD's to have a voice. We want to thought leaders, but we don't want like give people too much of an advantage competing with the money machine." That's my – No one in Google senior leadership has ever told me that, but that's my perception.

But you're right that AWS was always in search of a money machine. It was like, "We've developed these cool things and we're not making money in our real business. Maybe these can make us money." You can just see like a whole like rush to like just ship anything. Throw it against the wall and see if it sticks, because one day we had to produce money. Maybe that kind of lends itself to what we described earlier, like those motivations would put you in a world of like autonomy, figure something out and a world of like consensus. Don't rock the boat because we got to make sure that we preserve the money machine.

To that end, and you didn't ask me about this, but it would be interesting to see if those things change some. Google seems much more willing to pursue new projects when they feel like threatened. Yeah, guess that's how I would've phrased it, is Google fundamentally seems to innovate when it feels threatened. Amazon innovates out of survival out of like, I call it, that's just part of what it does to progress. But if you don't poke the bear, Google – It'd be interesting to see what Google does if they're not poked, because maybe there's not a lot of urgency to do something.

**[00:25:44] JM:** At least things that make money. I mean, there is some –

**[00:25:47] EA:** Yeah, they're always innovating on kind of like adjacencies around the money machine.

**[00:25:53] JM:** Like maps always gets better, and this and that, Android.

When you're inside Google, does it feel like the company is 10 years ahead of everything else or does it just feel like a total alien?

**[00:26:08] EA:** I think you're saying, "Is it weird or is it the future right?"

**[00:26:11] JM:** Right.

**[00:26:12] EA:** There's some implicit, like it's probably weird there. Is it weird because it's what the future is or is it just kind of strange?

**[00:26:18] JM:** Yeah. Some strange evolutionary diversion.

**[00:26:21] EA:** It surprises me. I mean, this is just one part of the Google internals. It just surprised me how much of what Google does. It's now the things the startups are all excited about. Like the travel and expense reporting. Like there's a whole bunch of wave of companies doing the stuff that like this internal thing called Trips that Google did that some engineer or a couple engineers came up with and it was ugly. But that's now how the rest of the enterprise is going to operate.

**[00:26:52] JM:** You mean there is an internal tool called Trips?

**[00:26:53] EA:** There's an internal tool called Trips, and you could – When you told it you wanted to travel, it would tell you the budget for such travel. If I'm going to Boston, this is about what it should cost, and we're going to give you credits to spend. If you don't spend, if you fly cheaply, you get to kind of accumulate credits for more expensive or last-minute travel opportunity. Then there's like half dozen companies offering those kind tools and doing well at them today that kind of move travel and expense approvals from – Democratize them. Like so much of old school enterprise was like asking your manager to do things and those expenses kind of – There no kind of central hub to like keep those in check. It was all kind of – So that's just one example. Another would be like OKRs.

**[00:27:46] JM:** I was just thinking the same thing.

**[00:27:47] EA:** Yeah.

**[00:27:47] JM:** The OKR and the KPI software.

**[00:27:49] EA:** Totally. That's like another thing that's everywhere.

**[00:27:52] JM:** It's like a product category now.

**[00:27:54] EA:** Yeah, the credit cards. Google gave me a credit card and I could use it anywhere. I didn't have to ever use like Conquer or Expense Software, and now you have Brex and Divvy that are basically giving you the same thing. Then on the infrastructure side, it was like containers were a thing and event driven architecture. It does feel like it's in the future, but it is kind of the – It's not obvious, because it's not – It hasn't been honed with like – It's not polished. Like Trips didn't feel like the future when I was experiencing it.

**[00:28:29] JM:** Was there anything you remember about the culture or about the internal tooling that has not permeated the real world outside of Google yet? Is there anything you particularly miss or can sense the absence of?

**[00:28:47] EA:** Well, there's the counter to that. Their elements, the Google holder that kind of got stuck behind, like Google Groups is still the primary mode of like communicating within Google. Everybody subscribes to a million Google Groups. Everyone's in email. Inbox was a mess and everyone's a mess. It's just complete – The volumes of email is out of control, because everyone – To get to this consensus building, everyone's subscribing to everyone else's team thing and they're all sharing with each other. But it's all in email, which feels just totally dated, and you can see how Google [inaudible 00:29:21] Slack, for example. Even the competitive response to Slack felt kind of slow, like, "What is this noisy thing?"

**[00:29:29] JM:** It hasn't come to market yet, right?

**[00:29:30] EA:** Yeah. The Hangouts chat I think is in market now.

**[00:29:35] JM:** Is it?

**[00:29:36] EA:** I believe so. Yeah. Here I am like naming secrets. No. I think we could Google it and we'll edit this show if it's not.

**[00:29:44] JM:** This is like the Microsoft teams kind of thing.

**[00:29:45] EA:** Yeah, it's Microsoft thing saying it came out of the Hangouts brand instead of being kind of one to one, and group chat was kind of threads and channels. That's the counter to what you're suggesting.

As far as like are there still like ways in which Google is ahead of the game and living the future? Probably. So they had this awesome internal – I don't know. I guess that's kind tackle too. They had this thing called Teams that was like who reports to who and what you're working on? I don't feel like that's been kind of well-permeated in startup land yet. What's the org structure? There're some org structure startups and there's other startups doing that kind of what am I working on and what's the homepage that I go to and I kind of see news within the company. There used to be intranets in SharePoint and I feel like that's kind of like Microsoft's era, and I feel like that still unresolved.

**[00:30:40] JM:** Nascent.

**[00:30:41] EA:** Yeah.

**[00:30:42] JM:** The internal company's social network. That is not a thing. That isn't a product.

**[00:30:46] EA:** Yeah, and there was this whole folly where we thought Facebook and Google+ were going to like become our internal social networks, and that never –

**[00:30:53] JM:** Could still happen.

**[00:30:55] EA:** it could still happen, yeah, but I think we lost to like five years of Silicon Valley innovation thinking that was the way it would emerge, and it hasn't quite.

**[00:31:02] JM:** How has Kubernetes changed the competitive dynamics of cloud providers?

**[00:31:07] EA:** Amazon's launch of the Kubernetes service was like an important point. Pretty telling situation. They launched – What is it? ECS? The Elastic Container Service? I want to say it even predated Kubernetes. I'm not sure. But it was kind of like –

**[00:31:25] JM:** About the same time.

**[00:31:26] EA:** About the same time. That was their Docker Swarm. That was their solution to the problem. Nobody liked it, or I'm being a little blunt.

**[00:31:35] JM:** No. It had some users.

**[00:31:35] EA:** Yeah, I'm sorry.

**[00:31:37] JM:** It was just proprietary. Nobody knew how it worked. There were like, "Look, we're just going to kick the can down the road. We don't know which open source projects are going to win. So we're just going to make a close source thing." Kind of brilliant.

**[00:31:47] EA:** Yeah. Right, which followed the like Kinesis, instead of shipping Kafka approach, and it followed other – That was kind of AWS's way of doing things.

**[00:31:55] JM:** Well, probably for different reasons.

**[00:31:57] EA:** Yes. Yeah.

**[00:31:58] JM:** I mean, Kinesis, they just – Like why do a Kafka solution when you could just totally wrap it in your own thing? But, yeah, similar philosophy.

**[00:32:06] EA:** Yeah. I think Amazon would have been pleased if ECS would have become the dominant way people run containerized applications on AWS, but that didn't happen. Customers said we're going to use Kubernetes, and they basically bagged Amazon to give them an EKS. To Amazon's credit, they did. I mean, I think that's – I applaud them for trying the proprietary, and I applaud them for then caving – Caving is maybe not the right word. For following the customer demand and being like, "Oh, you don't want this? We want this. All right. Here it is."

You asked how has it changed the way kind of providers operate. I guess that's just one example. The other is that I think as long as EC2 and images and VM, like the primary way of running applications, I think Amazon's got the first mover advantage and the upper hand. In a world where Helm charts or Docker images or other things, I think it abstracts the APIs that we've come to rely on for Amazon a little bit and it diminishes the portability and the lock-in, and I think that's super threatening to Amazon in some sense. But it's not game-changing. I mean, they still have – I don't think it's any reason to leave Amazon, but it's step one. It opens the door to leaving. Then if other people can continue to advance new things, now there's some liquidity in moving around cloud where there wasn't before.

**[00:33:41] JM:** Now that you're a venture capitalist, what are the investment opportunities in the Kubernetes ecosystem? Because by the way, I remember talking to you at the most recent KubCon and you're like, "I'm not sure what I'm looking for here."

**[00:33:56] EA:** Well, to be honest, I mean I may have said that, but what I was reflecting is that it's been a bit of a – Somewhat disappointing. I mean, I've talked to other VCs who looked at this space, and outside of a couple breakout companies, it hasn't felt like the – With the way the enterprise is talking about Kubernetes, you would've expected billion-dollar companies, several, to emerge.

We have course had Heptio, which had an awesome exit. I didn't seem like the business was taking off. So even then, even Heptio felt like here was like a really awesome team, super valuable VMware. Makes total sense. Great investment bet. But like was an indication that Kubernetes was this big startup business opportunity? It felt like it's been less of that than I expected.

**[00:34:45] JM:** Yeah. Now, it does make distributed systems easier to build and deploy and give to companies to consume. Is there some downstream impact that –

**[00:35:03] EA:** Yeah. Yeah. Certainly, there's – I was referring mostly to like pure play Kubernetes vendors as feeling like – And maybe all that business kind of just went to the cloud providers, and maybe it's just we were a little early and there's still wants to come. But the like the surrounding, the tooling that surrounds Kubernetes and microservices that have kind of emerged alongside Kubernetes and maybe cloud native, more broadly, has been just a boon for investment. I mean, every facet of tooling I think has turned over into kind of a next gen opportunity. There was a monitoring company and now is there monitoring company for kind of Kubernetes, microservices and cloud native? There was deployment approaches, and now is there deployment approaches for Kubernetes, microservice, security? There was security before when we were on VM's and now there's new approaches to cloud native.

**[00:35:57] JM:** But the question is are those approaches wanting entirely new companies or are they just alternative products built into Datadog or built into some cloud provider adjunct?

**[00:36:09] EA:** Yeah. I think that there's – I think. I mean, how can I make that a more interesting statement? There's already some evidence to that, and we invested in Honeycomb recently, which is like an observability company. You could tag observability as kind of like next

gen monitoring for cloud native if you wanted. It may not be – Charity might not agree with me with that tagline.

You see New Relic who's kind of the former generation doing things to more observability, more tracing, for example. But I still think – I don't see New Relic consuming all that space. I think there's 1, 2, 3 companies that could emerge there. I think that's true. Certainly, security running. Maybe it's not just Kubernetes. I think it's more broadly containers Kubernetes and microservices that are presenting, because those things have all kind of happened. Nobody's running containers. Kubernetes is how we're running containers. So I think the container security companies are becoming Kubernetes security companies.

I think if you just had VM security companies of Yonder, of or –

**[00:37:26] JM:** [inaudible 00:37:26].

**[00:37:27] EA:** [inaudible 00:37:27]. I think there's an opportunity it could be with them and build a big business.

**[00:37:31] JM:** What's been the hardest part of transitioning from doing product development to investing?

**[00:37:39] EA:** Somebody warned me – I don't if this is the hardest, but it's an interesting situation that somebody warned me about. They were like, to PM, you're always looking for like opportunity, but opportunity you can fix. It's like, "Oh! If we just tweak this week, if we can –" It's easy I think as a VC to want to buy a fixer-uppers as a form of product person and be like, "Oh! There's a bunch of opportunity here." If this company could chase it. If we do a little more of this and position it that way. Where I think the primary mode really should be like this thing is working. This is what the market needs. Certainly, they're going to run into issues along the way and you're going to have to address those and work with them to fix it. My experience I think can be helpful there.

But I do run sometimes the risk. I catch myself at times being like, "This isn't working, but we could make it work." I realized that I'm just like – As a board member investor, I'm like a part-

time contributor to this thing and there's already a dozen people working on it. Is my small contribution going to turn this thing around such that it's a valuable investment? I think that's a kind of a risk product managers face becoming investors.

The flipside is that, also, there is some aspects – There's a bit of learning to do around finance that I didn't come with and then I felt like there were others in the industry who knew much better than me. Fortunately, I think staying in infrastructure which has been my focus mostly has – I feel like my experience has been more valuable than my lack of experience has been the negative, if that makes sense.

**[00:39:18] JM:** Right.

**[00:39:19] EA:** Maybe one hard thing is I think sometimes it's easy to look too far into the future. I think at Google, it's hard to not look – There's no risk to looking too far into the future at Google. Eventually the future comes and the company has more has more products and tooling to absorb it than –

**[00:39:38] JM:** Google is not running out of solvency.

**[00:39:39] EA:** Yeah. But as a VC, you got to time your investments, and I think it was really – I feel comfortable with like, in some cases, what the future is going to look like. I quickly wised up to the fact that it's not only important what the future is going to look like, but when it will look like that, which is a dimension that was less important to me as a product manager.

[SPONSOR MESSAGE]

**[00:40:08] JM:** DigitalOcean makes infrastructure simple. I continue to use DigitalOcean because of the low friction and attention to user experience. DigitalOcean has kept the experience simple and I can spin up a server in less than a minute and get high quality performance for a low price. For an application that needs to scale, DigitalOcean has CPU optimized droplets, memory optimized droplets, managed databases, managed Kubernetes and many more products. DigitalOcean has the flexibility to choose the right instance for the right workload and he could mix-and-match different configurations of CPU and RAM.

If you get stuck, DigitalOcean has thousands of high-quality tutorials, responsive Q&A forums and a customer team who treats customers respectfully. DigitalOcean lets developers focus on what they are building. Visit do.co/sedaily and receive $100 in credit over 60 days. That $100 can be put towards hosting or infrastructure and that includes managed databases, a managed Kubernetes service and more.

If you want to get started with Kubernetes, DigitalOcean is a great place to go. You can use your $100 to start building your distributed system and you can get that $100 in credit for free at do.co/sedaily.

Thank you to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

**[00:41:44] JM:** You started a podcast called –

**[00:41:45] EA:** Yeah. All good people do.

**[00:41:47] JM:** So I hear. It's called Contributor. It's about open source maintainers. What are the themes that have come up in your conversations so far with open source maintainers?

**[00:42:02] EA:** One is the people and community – We use the word community, but I don't mean it in the way I used to mean it. But I mean it now like relationships. Just how much motivation of this is around relationships, which I thought this would be like a business podcast largely. These open source things have business elements to them and they create a lot of value for the economy. Then when I talk to these maintainers, I find out that a lot – Yeah, this is the way of interacting with people. One of our portfolio company's Chef, I talked to Adam Jacob at Chef and we got to talking about like – I don't know. I was asking him like how valuable the community was in this – And he's like, "No. These are people. These are my people. These are my friends. These are the people I woke up and had coffee with in the morning and spent time with. We talked about our kids."

To him, this wasn't like an asset or a collection. It's not partnerships. It's not BD. That was really awesome to hear. When we talked about what his future like, will he do more open source? It's like, "Oh, yeah!" and like the Chef project will live on in part because we are friends and that's not going anywhere. Even if like the company goes this way or the specifics of the project go this way or that way, licensing changes. What the world doesn't see is that there's this close-knit group of people who will care about each other. He's like, "That's personally super rewarding, and a lot of people aren't aware of it." So that's been a surprising element that came out of the podcasts.

**[00:43:49] JM:** In the episode you did with Matt Klein on Envoy, his description of interacting with the community actually sounded very much like the process of starting a company. I mean, his process of building the community and doing documentation and communicating with people, doing pull requests, making minor bug fixes. That was an incredible grind for him. In that show he described it as – It sounded like the most difficult work period of his life. The ironic thing of course being that he's not building a company around Envoy.

Do you have any reflections on that one? Because that one is particularly interesting. Envoy has been so successful, but it is not a product. It's not an open source project that's going to be productized.
It's not going to be turned into a company.

**[00:44:48] EA:** Yeah. I didn't describe it or I didn't internalized it the way you did, but I agree with you, that it did have – I remember, we had a conversation around when Google showed up to the Envoy project. They were super excited about it and he's like, "I had 10 Google engineers in the room with me and I felt like I was being acquired." I was like, "Wow!" Again, this like kind of business parallel of like he's this calm, scrappy. I don't know if that's the right word. But he's this kind of solo, if not, small group of people whose build this thing and then suddenly Google wants to show up and influence and own it or do something with it, and he's feeling worried that they're going to take his baby. Totally felt like kind of entrepreneur situation, and yet it wasn't. It was this other kind of parallel universe of open source, which is intriguing.

I applaud Matt for kind of taking a position on – I think if I were in his position, I think I would've struggled much more with the company starting decision. I can imagine myself kind of starting one and then kind of wanting to still do open source, and then he didn't seem to waffle, or at least his waffling concluded with some definite choice at some point that he's held to since then.

**[00:45:58] JM:** Yeah. Although it hasn't stopped other people from productizing Envoy, at least in the case of building service meshes around it.

Give me your survey of the companies that are productizing Istio.

**[00:46:14] EA:** Well, or Envoy.

**[00:46:16] JM:** Oh, yes. I mean, are there companies that are productizing Envoy that are not productizing through Istio? Istio being the service mesh built around Envoy.

**[00:46:23] EA:** Yeah. Yeah. There was a company that was closely affiliated with Envoy. It's no longer – In fact, Matt and I spoke about it and he was a big fan of it, and I forget its relationship with service mesh, but I don't think it was a service mesh solution and the name is escaping me at the moment and it doesn't matter in some degree and the fact that is no longer a thing. But Matt was like kind of like an adviser of the company. So to our discussion earlier, he was doing some commercial things, although it wasn't his company, and he was clear about that.

Maybe just to comment on that, Matt seems great about just sticking to his like welcoming all the people – If somebody launched another service mesh company on Envoy, he would be ecstatic, because that's a win for Envoy kind of thing, where if he was company founder, he wouldn't feel that way probably.

There's Solo, Idit Levine's company, that is I think – I don't know how closely. It's certainly closely tied to Envoy. I don't know how closely tied it is to Istio. They're doing things around increase interoperability I think with other service mesh-like implementations. I think they've explored whether they should also support other proxies, but certainly from the beginning, it was Envoy and mostly Istio.

There's Istio. Varun at Istio, founder, started the – Is it Tetrate?

**[00:47:45] JM:** Tetrate. Yeah.

**[00:47:45] EA:** Yeah. There are companies doing security. Kind of the service mesh layer. Isovalent is one that comes to mind. There's – And Calico kind of I think has – They also do security. I want to say some of it as at the service mesh layer.

**[00:48:02] JM:** I think Google itself.

**[00:48:03] EA:** Then Google itself. Totally. Closely related, but not Envoy or Istio, would be other service meshes. Aspen Mesh. Actually, they may be doing some things with Istio. Then there's the Linkerd the project and Buoyant, which in some cases predates all this and they're doing great things as well.

**[00:48:26] JM:** Is this stuff validated? Is the service mesh idea validated by the market?

**[00:48:36] EA:** To a degree, the kind of Silicon Valley companies who like – Lyft is all about Envoy and service mesh. If you're building on Kubernetes, it's feels like a valuable thing. But what's interesting to me is I feel like the enterprise is bought in to Kubernetes. They're not quite using it in production, but they think – Not only they think they need to, but they want, and they're like actively marching towards that. We need containers. Therefore, we run containers. We need Kubernetes. But I don't get the sense that service mesh is certainly at the same level of adaption. It's not like we need Kubernetes in a service mesh in the next year. It's like we need Kubernetes next year and we're looking at service mesh. I don't think it hit broad validation yet, but it feels promising and it's following in the footsteps Kubernetes did. There was a time when Kubernetes didn't have broad validation.

**[00:49:26] JM:** You did do a show about Istio, right? It hasn't aired yet?

**[00:49:28] EA:** Yes. Yeah.

**[00:49:30] JM:** With who? Was it Louis?

**[00:49:31] EA:** No. He's Swedish, Sven. Google engineer.

**[00:49:36] JM:** Got it. The reflections on Istio inside of Google.

**[00:49:42] EA:** Well, he was more part of the founding team. Mostly, my episodes are mostly about the story of how you went from thinking you want to do this to today. What's the kind of lifecycle of the open source project? With him, it was about how do you kind of get Google to give you a team to do this? How do you convince people that this is important? Then him discovering Envoy is like a foundational element showing up with Matt Klein and kind of integrating that in some discussion around how Istio – I think it's well-acknowledged kind of over-promised. I thing like launched a website –

**[00:50:20] JM:** Oh, you talked to him about that.

**[00:50:21] EA:** Yeah, we can do all these things, or we want to, or we plan to. It wasn't in version one, or two, or three.

**[00:50:30] JM:** It was like an ICO whitepaper.

**[00:50:33] EA:** Yes. Yes. Crypto mesh. I think he acknowledged – If I remember correctly, thinking maybe we did a little too much there, but also he felt like that framed the mission for both the internal team and the external team. It was like, "Here's what we are and what we're about," which I think is an intriguing way of – If the codebase is going to open, why not kind of a large thinking?

**[00:50:56] JM:** Well, Amazon has that press release thing. If you're going to create a project within Amazon, you start with the press release, which is like you write what the press release would look like before the product is really out there. Google took that practice perhaps a little too seriously in their development of Istio.

**[00:51:15] EA:** In this case – Yeah. It may just be – I don't mean to fault them, but I'm sure like a little tone swapping from like here's what we want to be, versus like here's what we expect it to be in the near term.

**[00:51:30] JM:** I thought it was like a problem of like the game of telephone. Probably, somebody on the Google engineering team was like, "This is what we wanted to do." Then somebody on the marketing team said, "This is how we're going to market it." Then people – The further it got downstream in the game of telephone, it reached more and more marketing people who said, "We should emphasize these points more and more and more." Then eventually it got to perhaps the IBM marketing team, because they were doing this launch with IBM. Then in each hop on the game of telephone, it just amplified these things that Istio just didn't accomplish yet. Then the people who are actually implementing it in the community or trying it out in the community saw, "I can't even bring this thing up." So it a huge disjunction, which was interesting story. I probably came out a little too hard on it. I was pretty critical.

**[00:52:33] EA:** Oh! In like one of your episodes?

**[00:52:35] JM:** A couple of them.

**[00:52:36] EA:** A couple of series.

**[00:52:37] JM:** You did a show about ClickHouse.

**[00:52:39] EA:** Yeah.

**[00:52:40] JM:** But hasn't aired yet. Coming back to our discussion of data warehousing, this is a column store database. ClickHouse is an OLAP database. There're a number of different column store data warehouses. What's different about ClickHouse?

**[00:53:03] EA:** There is a variant of the column store there was all about scale. Scale at the sacrifice of a little bit of speed. BigQuery, originally the Dremel paper, was kind of maybe one of the first descriptions of a column store. Dremel being Google's internal name for what is now BigQuery externally. If you run a BigQuery query, it's going to take at least a few seconds,

because it's not intended to be like a real-time, like, “I need to know the answer now.” It's like, “I want to get an answer out of a lot of logs and I’d like to do it while I'm sitting here. I don't want to like leave my desk and go get a cup coffee,” and to be able to interact with it.

That was what BigQuery or Dremel did so well, was like, “I can compute against maybe an infinite amount of logs and get a reasonable answer in under a minute,” and in some cases on the order of seconds.

ClickHouse and maybe Druid and some of the lag time series DBs I think are pushing that to like milliseconds, which is like – I'm not sure the limits on-scale relative to the other types of column stores, but getting you answers, sub-second answers, is kind of a new frontier in columnar databases. But ClickHouse is really well on that front. It's also interesting just that the origin story is so non-Silicon Valley.

**[00:54:28] JM:** What is it?

**[00:54:29] EA:** Oh, it’s Yandex. So the Russian search engine. They had – There're a lot of parallels between Google and Yandex and they have a Google analytics-like service I think called Yandex Metrica, and that team needed to build something. They saw the Dremel paper, but they didn't have the codebase to kind of built their own columnar store, but they were pushing it these kind milliseconds times.

This thing has been hardened and incubated inside of Yandex for – It’s like a decade old in some ways, and they eventually open sourced it. So the open source has only been available for a couple years, but it's different and that it's not like we open sourced it from day one and it had some rough edges, but we’ve improved it is a community. It's like here's an awesome database that’s actually operationally maybe even better than Druid, even though it's similarly fast. Operationally, meaning like you just kind of click deploy and start querying.

That's kind of – Like I said, I think the codebase is still owned by Yandex. I expect that they'll do some kind of foundation, and in our podcasts, the creators alluded to that. We’ll find some way to kind of make it more community-governed.

**[00:55:44] JM:** Are they doing a company around it?

**[00:55:46] EA:** They are not. No. No. There are companies emerging around it, but I didn't get – Yeah, the founders don't seem like they are.

**[00:55:56] JM:** Do you have a sense of the size of the data warehouse market? Is it big enough to support all these different companies?

**[00:56:05] EA:** Snowflake, maybe the biggest infrastructure enterprise exit we've seen in a while. The growth that Snowflake is just astounding.

**[00:56:15] JM:** Can you quantify it for me? Do you know off the top of your head?

**[00:56:19] EA:** Rumors are that it's like – I want to say like hundred million and doubling. More than doubling at that size, which is in revenue, which is like really unusual. If you look at other – AWS is like one of the few examples of companies that double north of a 100 million, or more than double north of a hundred million. Uber is in that camp or something.

Yeah, who knows how these things will go? I don't mean to predict it. The growth may slow. What's going to happen to IPO and multiples, what have you? But data warehouse, at least Snowflake – Snowflake feels like in some ways it's not the whole market, right? There's plenty of Red Shift going on. There's a bunch of BigQuery. There's a bunch of – They're just one aspect of the market. It feels like I mentioned earlier that some of the money that was going in the Hadoop – Big data has kind of moved making data warehouse surprisingly big market.

**[00:57:15] JM:** Yeah. Well, it also seems like Hadoop as successful as the Hadoop vendors were. In some ways they might've been too early. Because there's – I don't know what the words of choice were to describe digital transformation 10 years ago. But now we have digital transformation, which depending on whether you see it as something new, if it's something new or if it's just more proliferative than whatever digital transformation was 10 years ago, that significant increase in the market of people willing to buy a data warehouse product and, obviously everywhere, the volumes of data that people have are just bigger.

You would expect the data warehousing businesses to do better than ever. So you need fewer and fewer customers presumably, although the cost of storage has gone down. So maybe that countervails it a bit.

**[00:58:15] EA:** Agreed. I think digital transformation in machine learning, I would throw machine learning. Where before I think it was like before digital transformation and before machine learning, the problems with big data was like there's insights in there. If you just put all the data in one place and you start querying it, there's going to be things that will emerge. They're going to bring your costs down. They're going to save you money. It was all kind of vague and aspirational and unclear. I think the move the digital transformation feels like it's driven by like efficiencies, cost improvements, agility and it's kind of top-down. Machine learning just feels like this imperative threat that everyone's like, "We got to figure this out." Suddenly, like we're going back to doing what people said we would do before, but with some real urgency rather than just aspiration.

**[00:59:03] JM:** Does podcasting help you become better investor?

**[00:59:06] EA:** I feel like asking Adam Jacob if like his community is valuable and he's like, "No. It's just my people, my friends. I do because I love it." I don't know if it does. That's not the end or at least the kind of goal, but I enjoy IT.

I love about investing in part is meeting all these people who are doing amazing things and understanding how they're doing it and why. Podcasting is just another way to kind of scratch that itch, I think. Maybe it helps. I think there're some real market opportunities in infrastructure and in open source. So I think by shining a light on those, maybe I can find more opportunities there, because people know that I care about those things and see them as opportunities.

**[00:59:50] JM:** Speaking of the stories, you did a show with HY about a Alluxio. Alluxio is the system of distributed memory and storage APIs. He told the story of bring it to market from a university project. What are the lessons that you took away from the Alluxio story? More broadly, the trajectory of taking an open source project from academia to market.

**[01:00:23] EA:** So, as you probably know, Alluxio and HY incubated in the same like group – I don't know what it's called. Academic kind of unit as a Spark and other big-name data projects.

**[01:00:39] JM:** Mesos.

**[01:00:39] EA:** Mesos. Right. That was, in some ways, a well-trod path or at least it appears at the time. There're was a couple people ahead of HY. In some ways, it doesn't feel that. It feels kind of analogous to people who start Kafka inside LinkedIn or Kubernetes inside Google where you have this organization, institution, who wants you to succeed in the open source for whatever reasons they want you to succeed. It brings some brand equity to both the university and/or this big tech company.

Then you get to kind of – If it goes well, you get to leave and start a company around it, which is pretty awesome, I guess. Like you get to take the codebase with you because it's an open. You get to like build this project inside a company or university. I get a PhD and I get a startup. I don't think those were necessarily those people's motivations, but it is an interesting and effective path it seems to start to build.

In talking with HY, it seemed like he kind of knew from – He was like my thesis – My project was kind of this problem and he got two years to do it, which is a luxury that not many have with like super smart people around him to advise him on it.

**[01:02:04] JM:** One of the reasons I like to have investors on the show, there is a feedback loop between the investment community and the software development community. There is a sense in which the software developers inform the investors as to what their building. The investors have some sense of the overall market so that the investors can give feedback to the software developers in terms of this thing already exists, or you might be able to productize this thing in this direction.

The investors are repositories of knowledge and they become this kind of broad overseer type over the industry. As an investor, you have the difficulty of needing to survey a broad landscape and determine which areas in the landscape to go deep on, because you can't be just a survey or and be entirely broad. You have to eventually choose some areas to go a little bit deeper on.

Are there any particular areas that we haven't talked about yet that you are going deep on right now?

**[01:03:38] EA:** I'll just flag that we've talked about data engineering impart mostly because it's my background, but I also think that this is like – If you want digital transformation, you want machine learning to work in your company, you're going to have to solve the data problem. The data problem is surprisingly unsolved, like despite all the tools we've talked about, people feel like business users can't get the answers they want fast enough. Do I have a churn problem? I don't know. How do I find that out? I find an analyst in SQL query and they're going to go find a data engineer who can populate tables, and six months later it's still unclear. I'm bullish on the broad data engineering landscape in order to address the stuff we discussed.

Kind of off the wall, maybe one is that I think we built a bunch tooling around building software. Call it DevOps or continuous CICD, and it's changed the way we built software. I think the way we are going to build hardware will also change. The way we build hardware is largely the same as it has been for the last 20 years. You have like a designer who draws something and then you have like outsourced CAD users that fit into these tools and then that goes into like a simulation software. It goes through SIM, and as it breaks, it goes back to the designer and the engineer, neither of which may be CAD users. So there's also a CAD user in the room, and there's all these like functions. The CAD drawers, the engineers, the designers, the simulators in the same way that we had QA in ops and dev and SRE.

Part of what happened to DevOps is like a consolidation of functions, because we introduce automation to handle parts of the work. Then it was more easy for one person to kind of get the rest of it all done and have more ownership, which allows us to have kind of very fast timelines, if it's all in one person's mind.

I'm excited about engineering software, which is what we call it hardware land, which is a little confusing because it could also be software engineering software, but I guess mechanical engineering software as an opportunity. There are some other corollary trends that feed into that, like 3D printing and additive manufacturing. In a sense, that's like the cloud – I think we always wanted to do continuous integration with software, but there was no impetus to change.

We are still shipping software in a package, shrink-wrapped for some time. Why build daily or hourly if you can only deploy once and you already have these kind of atomic kind of intrusive updates?

But with the cloud, that was kind of the tipping point that said, “Okay. Now we really should change, because we can,” and three printing additive manufacturing I think does that where like maybe having a six-month build cycle on design is too long if we can print it in a day. If you can print it in a day, maybe we should have a build cycle in a day. That's one area I'm excited about. This is maybe off the wall, or at least – I think there's also – This is maybe somewhat well-understood, but I think there’s also an interesting opportunity around a pattern of there used to be services businesses and investors didn’t like those. They’re kind of like lots of labor, low margins, and we want software businesses, highly scalable, high-margin.

But there's some path to get to like automating all these things to your machine learning, and part of that path is that we need people to do the job first. Then we need to instrument those people to understand what they're doing and then we need to use machine learning and software to automate what they were doing from what we learned from the instrumentation.

I think there's a path for companies to like we’re going to hire a bunch of people to do this task for the next two years and then we’re going to automat it overtime and then we’re going to go public or exit at high-margin software, which is kind of a compelling model and it feels like if we’re to cross the barrier towards these automated things through a machine learning, it seems like a viable path.

**[01:07:49] JM:** Is this the thing that people call robotic process automation?

**[01:07:52] EA:** It's not. Robotic process automation I think is like we’re going to sell you software so that your company – So it's related. Your company can automate their tasks that you're doing already. You’re creating invoices. You’re underwriting loans, which is kind of a bunch of repeated manual steps and we’re going to stick a shim of software between you and the keyboard and we’re just going to watch your keyboard strokes and then we’re going to replicate them. It’s like Excel macros across the desktop. That’s robotic process automation, poorly named or otherwise.

What I'm referring to here is like if we wanted to automate QA today, software QA, we need someone to write – Or test writing. No one likes writing their software tests, and maybe they're predictable as to what you need to write. One way to get there is to hire a dozen test writers, or let's take accounting, someone to keep your books.

I can imagine there being eventually an automated accounting company, and until we're there –

**[01:08:54] JM:** God. I hope so.

**[01:08:55] EA:** For the next 10 years, maybe we can give you like an outsourced accounting startup that can evolve into an automated accounting startup. Then maybe I'll just tag onto that. I think that there's probably bunch of software that needs to – If you do one of those startups, you have to build two products. One is the like service you give your customers, and then two would be the instrumentation product that your laborers are using that observe your laborers and what they're doing. I think there's an opportunity to build software that instruments knowledge workers.

**[01:09:29] JM:** Eric Anderson, thanks for coming on the show.

**[01:09:31] EA:** Thank you. It's great to be here again.

[END OF INTERVIEW]

**[01:09:42] JM:** Being on-call is hard, but having the right tools for the job can make it easier. When you wake up in the middle of the night to troubleshoot the database, you should be able to have the database monitoring information right in front of you. When you're out to dinner and your phone buzzes because your entire application is down, you should be able to easily find out who pushed code most recently so that you can contact them and find out how to troubleshoot the issue.

VictorOps is a collaborative incident response tool. VictorOps brings your monitoring data and your collaboration tools into one place so that you can fix issues more quickly and reduce the

pain of on-call. Go to victorops.com/sedaily and get a free t-shirt when you try out VictorOps. It's not just any t-shirt. It's an on-call shirt. When you're on-call, your tool should make the experience as good as possible, and these tools include a comfortable t-shirt. If you visit victorops.com/sedaily and try out VictorOps, you can get that comfortable t-shirt.

VictorOps integrates with all of your services; Slack, Splunk, CloudWatch, DataDog, New Relic, and overtime, VictorOps improves and delivers more value to you through machine learning. If you want to hear about VictorOps works, you can listen to our episode with Chris Riley. VictorOps is a collaborative incident response tool, and you could learn more about it as well as get a free t-shirt when you check it out at victorops.com/sedaily.

Thanks for listening and thanks to VictorOps for being a sponsor.

[END]