0:01

Alexey

**Some people call you the “Father of DataOps”.**

0:04

Chris

[laughs]Oh, yeah. One guy called me the “Grandfather of DataOps”. That hurt. [laughs]

0:13

Alexey

**I'm just thinking about the introduction – should we use that. Or do you prefer I don’t? [laughs]**

0:18

Chris

Whatever you want. [laughs] Yeah. I've been doing it a while. So yeah – whatever term you want to use.

1:20

Alexey

**This week, we'll talk about DataOps. We have a special guest today, Chris. Chris is the CEO and Head Chief at DataKitchen. Some people call Chris the “Father of DataOps” maybe he will tell us why. Welcome to our event today!**

1:38

Chris

No, thank you. Yeah, thank you for having me. I'm happy to be here and tell some stories. So get a blanket and some hot chocolate. And hopefully you won't fall asleep.

# Christopher’s background

1:50

Alexey

**[chuckles] I'm sure nobody will. Before we go into our main topic of DataOps, let's start with your background. Can you tell us about your career journey so far? Maybe you can also mention why they call you the Father of DataOps?**

2:01

Chris

[cross-talk] Yeah, that's fine. So the headline is sort of “Old Data Nerd,” I guess that's the headline. I grew up in the central part of the United States – Wisconsin – where there are a lot of people of German heritage. My last name, Bergh, probably is evidence of that. So, I worked my way through college and then I spent 15 years building software at companies like NASA, a lab at MIT, some internet startups, one called Microsoft, and then I sort of got the management bug.

And then in about 2005, when my kids were young – actually, I think my son was seven – I was like, “Oh, I need a break. I want to be home by five. I'm a big software guy, I'm gonna do this data stuff full time. Because I've done a little of it, it'll be easy. No problem.” And it wasn't, actually. So I managed people who did data science, data engineering, data visualization, and my life was bad. It was just… things were breaking left and right and we couldn't go fast enough. People always had more insight questions and my team wanted to innovate. So, in some ways, I've been trying to solve that problem now for the last… many years.

In the interim, I think, I've really come to realize that the problem is less – because as an engineer who codes and has written a lot of code – it's less about the production of code, and more about the processes around the building of the work. It really is that realization, back in 2005 and 2006, that I ran a factory – data came in on one side, and there were places where that data was assembled, and models were applied, and results were added to it. How to run a good factory was not something I knew as a software engineer. So I had to go back and read about industrial process automation and Deming and Lean techniques.

4:12

Alexey

**Was it an actual factory? It wasn't a metaphor, right?**

4:15

Chris

No, no. Well, it works as a metaphor. In some ways, I think, when you have a lot of data pipelines in production, and you have customers who will yell at you when they're late or when things are wrong, having a good production process becomes really important. But then on the other side – you run a factory, so you have a hard hat – but on the other side, you really run a software team. Because changing the work that's in production, the rate at which you can change, the cycle time at which you can change, is really an important determinant of customer success. Because you don't want to have an idea and then spend three months working on it and then having your customer go, “Eh. That's not quite it.” [laughs]

So, you run a factory, which is where you need a hard hat, but you also run a software team that has to be agile. So how do you do both those things at the same time? You kind of have to deal with hipsters and hard hats with cycle time and error rates in production. We started this company about eight years ago and those issues were really burned into my brain as a leader, like “Solve those.” No one understood what we were talking about then, in the sense that this idea of automation, and Lean techniques, and DevOps, was really very, kind of “out there.”

We had to talk and make a word for it – we called it “Agile Analytic Operations” for a while and “DevOps for Data Science”. And then about five, six years ago, we picked DataOps – we made a list and picked DataOps because it was the shortest possible name. It had been in use before and we wrote the Wikipedia article, and a manifesto. Over the years, we've had to write quite a bit and talk about it, just to get this idea across about the “how your team works,” rather than “what you do.” I mean, what you do is cool – like model selection, or data transformation techniques and visualization – those are all really good things. The basic idea of DataOps is, “if you build a system around that – that automates a lot of the monitoring, deployment, collaboration – your productivity goes way up and your customers are much happier, and you end up doing better work.”

# The essence of DataOps

6:42

Alexey

**Is this the essence of DataOps?**

6:45

Chris

It's the essence, yeah. It's about the system. It's about *how* you do things rather than *what* you do. It's about the assembly line and not the individual assembly.

6:56

Alexey

**So is it more about tools? Or more about the approach? Or both? Maybe you can tell us what DataOps actually is. You said it's a system that automates all these things like monitoring, that helps people be more productive – so people don't need to worry about broken pipelines and when things get broken, they know how to fix it. Is that right?**

7:22

Chris

Yeah. I think the focus is really on three things. 1) Lowering errors in production. What do I mean by that? I mean – if you follow data from its source all the way to where it creates value, including all the different tools and systems. The metaphor I use for that is a factory, right? Where you take it and put it in a bucket store, you put it in a database, you aggregate it, you run features on it, and build a model, it gets cached in a report, it's governed – all those pieces that go with it. That journey that that data goes on, metaphorically, is a factory process. You want to have a factory that produces good things – one that has high quality and low errors.

For us, we think focusing on error reduction means you improve customer data trust and if you notify about problems – and that's really an “observability problem,” if you want to use a DevOps term, or a monitoring problem, or a data quality problem – the industry hasn't quite named what it is. [laughs] But really, it's like “Check the stuff that you're doing.” An example of that is – about a month ago, I talked to a guy who's a data leader of one of the biggest companies in the United States. He got a call from the CEO of his company saying a report was blank. It took 26 people out of his team of hundreds to find the problem. It took them six hours. And it was a dumb problem, like a blank field, this, that – it's a dumb problem. But the point is, you should know if the results work.

Another story I tell is – in 1990, my first job out of graduate school was to work at NASA Ames. At the time, NASA had put the Hubble telescope in and it was blurry. Right? It didn't work. And no one had looked through it before it launched. No one had said, “Oh, let's go look at that house next door!” And I think that's very true of our data and analytics teams. We do all these little individual things and it doesn't work. So error rates in production, and then: 2) Cycle time of deployment. How fast you can get new models, new datasets, new visualizations, from your mind into production. And that's both velocity and risk. Then 3) is really focusing on team productivity, reducing the amount of meetings, and collaboration.

Then the last part is sort of a principle that these processes that you do – error rates, and cycle time, and productivity – they're measurable, and you should measure them. You should work on them. Which is surprising that we work with people who are very analytically-oriented, but these incredible, important metrics on their team success, like “How much work is your team doing?” “How often are things breaking?” “How fast are you getting new things into production?” Those are really important metrics. Overall, it's very value stream-focused. It's not so much about data science or data engineering – it's really about this Lean principle. You always optimize the whole and not the part. The reason for that is, obviously, in manufacturing, you take an individual workstation and you make it super-powered, there still could be a bottleneck later on.

The same thing applies to data and analytic pipelines. You could have the best model in the world, but it's fed by bad data, or it doesn't get in the hands of production, there’s going to be a problem. Thus, over-optimizing a part, both from a throughput standpoint, or the amount of insight it generates, seldom works. It’s a very holistic discipline and that also drives productivity. In some ways, it's an idea. We have a company and we have software that accelerates that idea. But just like Lean manufacturing is an idea, or DevOps in software development is an idea, there are tools to accelerate the implementation of that idea.

11:51

Alexey

**So, correct me if I'm wrong – you said the focus of DataOps is on three things. The first thing is – you want to reduce the number of errors you have in production. We can think of this as a factory line or some assembly line, and we want to monitor that everything is fine there.**

12:14

Chris

You want your factory to produce a Mercedes, and not produce like an American Motors car from 1975. [laughs]

12:22

Alexey

**Then the second thing is the cycle time of deployment, right? And the third thing is – team productivity should increase. Right? I imagine that if you want to have an effective factory, it's not only about the tools and the monitoring system you have, but also about how the people operate there. So it's both about tools and the processes of how exactly people use these tools. Is that correct?**

12:50

Chris

Yeah – error rates, cycle time, productivity, and then the last thing is measurement of those things. Because you can't improve what you can't measure.

# Also known as Agile Analytics Operations or DevOps for Data Science

13:05

Alexey

**Did you say that you actually didn't know what to call it? I think there were a few names, “Agile Analytics Operations,” then “DevOps for Data Science,” but then it was actually you who came up with this term – “DataOps”?**

13:20

Chris

No, no. The term was already there. There's a company in the United States that has been using “DataOps” since the 80s. I think the term… I guess my thought was that we're trying to get an idea across and we needed a name for that idea. I'm pretty arbitrary on what the name is, so I felt “the shorter, the better.” We could have called it “Tom” and I would be happy. But the idea has to be hooked up to a name. The reason is, when we went to conferences and we were wearing chef's hats and people were looking at us like we were aliens because no one understood what we were talking about.

It's frustrating to go and work for years and build software and then no one goes, “What? Are you an ETL tool? Are you a data science tool? Do you build a data lake?” I'm like, “Ah, no. We do this thing called DataOps.” “What's that?” [laughs] And I think at that conference – it was like a Strata conference, a big conference – we talked to hundreds of people and had all this (it's very frustrating). We finally talked to one guy, and I'm describing it to him in these terms, and then he goes, “Oh, that sounds a lot like DevOps.” And I was like, Yeah! Yeah, it's just DevOps for data analytic teams.” And he goes, “Oh! That makes sense. You mean people don't do that already?” [laughs] I was like, “No. No one. And that's the problem.”

There's a sense, if you're coming from a software development background, that automating and testing, and monitoring, and the processes your team works on, are really important and worthy of investment. And they're not just for lesser beings. If you can get those right – and actually, what's happened in software, the team percentage of people who are allocated to, instead of doing the work and working on the processes and automating around it, is upwards of 25% of the whole team. And their pay is actually sometimes higher than the people who are actually doing the stuff like JavaScript or building a web back end. If you get the system right, it actually allows you to make a lot of changes quickly. So if you look at a data analytic team, 1-2% people are doing it on weekends.

It's not seen as ‘worthy work.’ So that's partly my mission – that work is actually really worthy and you own the processes that you work on and you can control them. And if you don't, what happens is – your life sucks and it's not good. I've suffered for many years. I suffered in two ways – one is, I was a bad manager. I blamed people instead of the system. If you actually read Deming and the Lean principles – 95% of the time, it's the system or the process you work in and not the person's fault. As a leader, guess who owns the process that people work in? The leader. [laughs]It's much easier to say, “Oh, this person sucks, I'm going to fire them. They're letting us down.” I did that and I'm ashamed, actually, that I did that sometimes.

It's partly, also, when you are in a data analytic team and your customers are rolling their eyes, because they ask you something simple and you say it's going to take months, or your data providers are giving you crappy data, and you're just caught. [sighs] And it's not a fun job. I think there's a lot of evidence of that. Like, Gartner says “60% of projects fail.” There's the model – most models don't make it into production. There's our studies of the amount of errors that are happening. We even did a survey with Data World of the psychological state of data engineers and it was actually significant – we got 600 responses. So it was a decent, statistically relevant survey. 70% of them said their job was so stressful that they wanted a therapist to go with it. Actually, I kind of wasn't surprised.

It's not a great job, because once you get something working – you struggle to get something working – and then people ask you for more and more and more. And then things break left and right. Then, you're like, “This sucks,” and then you quit and take another job. [laughs]

18:14

Alexey

**Where you do the same thing.**

18:15

Chris

Yeah, you do the same thing again, and that sort of complexity and complications, and make your life not fun. I just don't think we need to live that way. I think there's a better way. And I think it comes from this thing – this idea that we, as people, own our process and we can take control and just don't have to suffer. We just don't have to live that way. Because I think it actually ends up hurting people's psychology. When it was my 42nd birthday – this was in 2007, or whatever – one of my data engineers had his 24th birthday. So he came in on a one-on-one and we're talking with him, and he started crying… Because he just felt bad. Like, “I can't go fast enough. Things are breaking. My customers are yelling at me.” And he was just like… he's a really good guy. And I felt like crap.

Because, here he was, taking it very personally – and actually, that happens to a lot of people. So, we have a backpack full of tasks, we've got data providers who don't care, we’ve got systems that are “funky,” and we're just told to “work harder.” You know? I think, if anything, the idea of DataOps is about taking control back and saying, “These methods that worked in manufacturing – these methods that worked in software – let's just steal them and apply them to our jobs.” That's it.

19:51

Alexey

**You have to own the process or the process will own you, right?**

19:54

Chris

Exactly. Exactly.

# Defining processes and automating them (defining “done” and “good”)

19:56

Alexey

**All these “funky” systems, bad quality data – that will keep you awake at night, right? You don't want to have that. You want to be in control of these issues, so you need to take control of that. Yes, so you said that having these processes is important, so we need to work on these processes and automate them. But how do we actually do this?**

**Let's say, I'm a data engineer – I just joined a startup and I need to build a data pipeline because the management wants to have a dashboard. I do this in Spark or whatever. It works. And then the management, of course, were satisfied and then they started giving me more and more and more tasks.**

**I don't have time to just slow down and then take control of this and make sure that it's not duct taped together. I want to be sure that it’s a solid thing that does not wake me up at night. How do I go about defining these processes and automating them?**

21:02

Chris

Well, I think it's the definition of success and the definition of “done.” So I think they're definitional questions. Running towards making your customer happy and being a hero is bad in and of itself. There’s the heroic act of bolting together Spark queries and dashboards– however you want to do it – integrating data and having the higher ups go, “Oh, thank you. That's great.” Then you get 10 follow-up questions and you're working on the weekend – somebody changed something on the data feed and you're fixing it on Sunday. So, I think that that's not bad, but what does “good” look like? And what good looks like is – if you've done something that you can run it, and it will tell you if something's wrong while it's running. Then, you'll be able to make a change somewhere and you'll know if there's a problem in it. *And* you can also hire a 22 year old who can start working on it and be able to make a change quickly.

So what is “done” and what is “good,” we have to talk about. Because a lot of times we think “done” means our customers are happy. And then it's on to the next thing. But then the discussion with the customer comes and it’s like, “Look, it's not done because of those situations.” I need to do more and so we need to set up a system to handle this complexity. I was a software engineer in 1996 and I was working on live websites – it was a very early sort of Facebook-ey website. We had 4 million people on it. I was restarting the server, kicking people off, I was there late at night. And that was cool. I was like, “Hey, you're a hero.” [laughs] Do you want big companies doing that nowadays? No. So about what's “done” and what’s “good” – I think you have to have a discussion with people.

Also, it becomes a question of trade-offs. Like, “Okay, I can go really fast, but I'm building some technical debt in the next sprint. Can I have some time to reduce that? Or else going to be able to do less in the future?” Right? “Can I have some time to refactor?” “Can I have some time to write some documentation?” “Can I just write some governance documents?” “We want to hire more people, so I want to put in more automated tasks and more automation around it – to make it easy.” It becomes a discussion of time allocation. Because if you're in that situation where your value only comes from the insight you've generated and not this discussion, it just becomes a harder life. Because you get buried and then unhappy, and then… I've seen it – people quit. A lot of people have entered the data and analytics field in the last five years and, actually, there are a lot of unhappy people. So it's because of this situation.

24:05

Alexey

**Okay. So, in this case, I think you partly gave an answer to that – as a data engineer who just developed this Spark pipeline and put together some stuff on a dashboard – do I immediately go to the management and show that? Or do I wait a little bit and then maybe think, “Okay, is it done? Is it ready to be shown?” And then maybe I think about the processes and think about “How can I make sure that this thing is actually reliable?” At what point do I go to management and say, “Okay, this is ready.”?**

# The balance between heroism and fear (avoiding deferred value)

24:41

Chris

DataOps is all about balance between heroism and fear. One way to do it is to have fear and sort of say,” I'm not going to show it to people for weeks and weeks, *while* I do all this stuff.” And…

24:56

Alexey

**I can see how that might end up not really good.**

24:59

Chris

It might not be good. Because all that work that you're doing – you may capture the requirements wrong. So, like a lot of things in life, it's about balance between heroism and fear. I do think you should get early feedback from your customers. And I think you should tell them, “Look, I'm giving this to you – I haven't checked the data perfectly. I haven't automated, I haven't tested it. It's not in version control. This isn't production-ready yet. So I've got some technical debt that I'm going to have to add.” You don't know – that may be good enough and then they're on to the next thing and they want you to throw it away. *Or* they may have six more things that are based on it. This may be the cornerstone that you can build the whole infrastructure on.

You don't know what's in your customer’s head and that's okay. It's okay to be humble and iterate your way into really understanding what your customer wants. Like, I've made this mistake – you have a lot of calculus, graduate degrees, “Man, I'm really smart. My business customer drank beer and went to business school, and I’ve got to work for him.” And you're kind of pissed off, like, “Man, I did all this calculus.” But the reality is – they know a lot of things that you don't. So we happen to have this strange talent of high abstraction, putting data together, and we like our beer, but we're not good business people. So, I think, giving them early bits of what you've done helps to make a dialogue with your customer on what is on their mind. What you need to do, partly, is that.

Our tendency as technical people is to build very complicated things – these sort of Crystal Castles. I stopped coding five years ago, and I miss it. I want to go, partly, sometimes I have to stop myself. Because it's like, I *want* to go off for like a month and code – because it's really fun. [laughs] But I've learned that, if I do that – I'm often wrong. Getting early feedback, it helps. We're really just trying to maximize the amount of value you bring to the organization. And that's what really matters. It's not infrastructure – it's not *deferred* value, it’s *value*.

# The Lean approach

27:42

Alexey

**I think this is one of the principles that you mentioned – this Lean approach, right? In Lean – I think there is a book called “Lean Startup” – you don't do that. You don't close yourself in a garage for a year and implement something that maybe nobody cares about. You actually show it to people, get feedback from them, and then iterate on this feedback. This is one of the principles behind DataOps as well? And DevOps?**

28:14

Chris

Exactly, yeah. It's a very iterative methodology that's based on finding the right balance between heroism and fear – between being humble about what you know and forcing feedback from your customers. It comes from Agile software and the Agile Manifesto – the DataOps manifesto we wrote is sort of a blatant rip-off of the Agile Manifesto. The problem with data is a little more complicated than DevOps, because we have two major cycles of iteration and not one. So, one is your iteration with the customer on “Is this data telling you what you want?” And “Am I showing it in the right way?” And then the second is, you've got an iteration with the data itself, like, “Is it actually predictive?” “Is this the right data set?”

So you've got these two cycles of iteration. In software, you sort of an application, you send it to someone, they go “No. Change this. Change that.” You've got one. You're really working on, “Can they use it?” And “Do they understand it?” And we have that in data teams, but it's also like, “Can the data actually say what is statistically relevant?” “Can it support the story that you want the application to have?” So it's a data problem *and* an application problem – and that makes it complicated. Also the value streams – the factory as data goes – is much more organizationally complicated. In bigger companies: you have an IT team, maybe a data warehouse team, a data lake team, you'll have a data science team, you'll have end-user business analysts, you have governance – and then over here, you'll have the person who's receiving the value. So you end up with these silos of optimization.

If you really believe in these Lean principles like I do, you need to optimize the whole. You need to optimize *across* all the teams. And you need to think about how “If I'm going to change one piece, its effect on the other.” That's what happens a lot of times – that one person who has one pipeline for the startup, the startup gets successful – they hire a data engineer, and a data scientist, they have business analysts on it, they hire some more. Pretty soon, you've got the data engineering team, and the data science team, and you've got a couple of different lines of business with business analysts on, and then you hire someone for governance, and then you've got three definitions of metrics in three different places. People are running around crazy trying to get stuff done, but you're still not achieving what you think you can. So it becomes a sort of balkanization. The separation of these teams actually makes people –which makes sense – but it's harder then to deliver iterative value. [cross-talk]

# Avoiding silos

31:23

Alexey

**You have to put everyone together, right?**

31:26

Chris

Well, people put blinders on. The data team worries. My job is to get value up to here [gestures with hand]. And I don't really know what happens. I've washed my hands. What does that data science team do? I don't know. And then the people who are doing visualization, who've taken maybe a scoring model and the data, they're like, “These people are too slow. These data science people don’t understand the business. I'm just gonna go make the segmentation in Tableau and change it and throw all their stuff out. I'm gonna go to the raw data and bypass all this stuff they've done.” So everyone's optimizing their part.

I've run into cases where teams have spent years working on something and it hasn't gotten to production, – large teams, teams of 400 people – they spent two years and nothing's in production. Nobody's receiving value. I've learned to desperately want to deliver value to people who use it. Sometimes those are people using Tableau. Sometimes those are business people. And if I don't, I get really worried that something's wrong. And that sort of instinct is what I want a lot of people to have. Unfortunately, it's not there – to be blunt.

There are a lot of people who are like, “Ah, I'm doing enterprise infrastructure and that's it. I'm building a data lake.” “Well, who's gonna get value?” “I don't know. We're gonna get all the data in the lake and I'm going to build it and it’ll come. Things will be great.” And then there are this sort of techno fetishists who are like, “Hey, we got a nice new tool. I went to their conference and had a great time. This tool is gonna accomplish everything for us. So if we just put all our data in this tool, good things are gonna happen.”

33:14

Alexey

**Hadoop, right? I think Hadoop 10 years ago was [cross-talk]**

33:17

Chris

Hadoop 10 years ago, the cloud now, “Put all the data in the cloud, good things are gonna happen.” You know, it's Hadoop, it’s Spark, it was Oracle before that, and cartridges and… it's a way to defer value. Deferred value as a trap. Like, “I'm going to build something really good and that's going to be generative.” It’s a trap that often hurts you.

# The 7 steps to DataOps

33:47

Alexey

**You described this picture of isolated teams – data analytics teams – who don't care about data scientists, and then there are data engineers who just create data lakes because they want to. If we go back to that one data engineer that everything started from – what could these data engineers have done to actually avoid all that? We talked about automation – thinking of processes and automating them – which things should that data engineer have automated to avoid all that? Not just the data engineer, but I guess the management also – what should have done to live happily ever after?**

34:37

Chris

Well, we used to give this talk called “The Seven Steps to DataOps,” so I'll talk through that. One of the first steps is, “Take your code and put it in version control.” Don't have it on your hard disk somewhere or file share. All the code – the report, the transformations. That's one thing. Then the second is, “Write automated tests that run in production.” You're expecting 1000 rows, you're expecting this report to have this value – write a test to check it and monitor it. The third is, “When you're changing something in development, run automated tests against that to judge regression or impact analysis.” If you change something on the back end, you’re able to tell if the front end is broken in a very simple way. So it's testing, it's automating deployment, it's version control – and then a lot of times it's counting when you have problems.

If you have an error, make a JIRA ticket or put it in a spreadsheet. If you want to improve your cycle time, just every week – start trying to deploy every week. And if you don't, note that. I think those things are good. Then you should also have this case of, “Can I hand this off to someone else? Do I have to own this for the rest of my career here?” There's a lot of times in these data processes where, like, “I got my Spark jobs, they run, and then I hit a button on my reporting engine to refresh.” Is that a problem? Well, no. And then you get these documents that say, “Do this on Thursday. Do this on Friday. Check this.” and they become these sort of playbooks.

You know, I'm sort of against checklists. I'm more part of the Anti-Checklist Manifesto. I'm like “Write code that automates that.” Everything that you're doing a checklist for, automate it away so that the system just runs without you and will tell you if something's wrong. The next thing is – you can hire someone just out of college and they can make a change in their first week without having to talk to you.

37:13

Alexey

**Is this DataOps?**

37:14

Chris

That’s DataOps. Or is it DevOps? I don't know, like – I don't really care. [chuckles] It's very focused on this sense of automation. Actually, it's badly named – it's more like, “Code that acts upon data operations.” [laughs] Because it's not really about the data, honestly. It's about the things acting upon the data.

37:43

Alexey

**You mentioned checklists. If you have a checklist – you don't like checklists you said – but it's probably a good first step, right? To have this checklist. [cross-talk] It's better to automate a checklist that exists rather than, you know, not automate anything.**

37:59

Chris

Absolutely. Yeah, absolutely. [cross-talk]

# Wanting to become replaceable

38:01

Alexey

**So that’s the first step for that data engineer, before they hire a college graduate. Right? To have some sort of system.**

38:08

Chris

Yeah. You want to have a vacation, right? You want to have weekends off. So writing a Wiki page that says, “Here's how you operate this thing,” is good. Then having that discussion with your customer saying, “Look, I know this is taking 10 minutes a week, but if I quit – you're going to be stuck and no one's going to be able to do this.” Or, “We want to hire some more people and you want me to do more things. I'm spending two hours a week on this production stuff and I should be spending two seconds or I should be spending *no time* and just have an email come.” So I think that the sense is that this is the idea that you're automating, and also that you're running towards errors. Inevitably, data systems break. That's just the fact of life. So, what you want to do is – when you have a problem, you want to find out where it is as fast as possible, and automate the fix to that. Tests and automation – that's a good thing. Some organizations end up in shame and blame culture, where errors happen and people don't want to talk about it. That's why I think counting your errors and tracking them is important. And I think there are a lot of organizations who have very poor data quality – they're missing their SLAs, no one trusts the data – but they're claiming great success. [laughs] It's like, “No, it's not working. Maybe you should just have a report that actually has those things in?” Anyways, does that help?

39:46

Alexey

**Yeah. I guess one of the things you mentioned, which is a key here is – as that only data engineer, you don't want to be indispensable, right? You want to be replaceable. If you’re the only person who knows how the system functions, then maybe in the short term, you're good. You can ask for a salary raise because they're not going to fire you.**

**But in the long term, it's not really good. Because you want to have vacations, you want to relax on weekends. So you actually *want* to be replaced and all these things that you mentioned, like automating, checklists, and all the other things – they really help.**

40:29

Chris

Yeah. I think it's not wrong, emotionally, to want to be indispensable. I spent a lot of my twenties wanting to be indispensable on a software team and wanting to be the “studly guy who built the cool thing,” and get the praise from other people. I was good, you know, and I was like, “I'm gonna build the cool thing!” And that's not wrong. It's normal. But you don't want to build such complexity that you have to own it for the rest of your career, or that no one else can take it over. We have software-developed terms like “technical debt.”

My favorite term is “hairball”. Like a big hairball of stuff, like from a cat that’s throwing up a hairball. I actually think that that's sort of true – you can very easily create a hairball that no one else can untangle. Thus, your definition of what's “good” and what's “done” has to change. You're not “done” when you've created a hairball. It's not “good”. I also think when other people do that on your team, you have to go “Ew! I know all the business people are loving you, but you just coughed up a huge hairball. And that's not good!” [laughs] “You're not done and it's not good!”

41:53

Alexey

**Do people like you after they hear that from you? [laughs]**

42:00

Chris

Eh. I think there's nice ways to say things. There's business language, like, “Let's talk about what ‘done’ means and let's talk about what ‘good’ means. Okay? Yeah, you did this great, but you're not done and it's not good. Because it can't be handed off, because it's not version controlled. Let's write it down and let's agree about what ‘done’ and ‘good’ is for our team – with our customers.” I think there are methods. I'm being facetious on the emotional stuff.

I do find that, as a manager and a leader, I've had to publicly praise and privately criticize people who are heroes, like, “Hey, you worked all weekend. Fantastic.” And then like, “Okay, you worked all weekend…” Sometimes we have to do that. Sometimes, you *do* have to work all weekend. But that should be rare. And sometimes you do have to create hairballs. And that's okay. But you shouldn't do that 95% of the time.

# DataOps is doable

43:06

Alexey

**All these steps you described that this particular data engineer could take to improve the system – they don't seem to be that difficult or that complex. Okay, put the code in Git. I think everyone's doing this – I hope. Then tests. Okay, this step is a little bit less easy, because nobody likes writing tests, but it's doable. Then integration tests – you want to be certain that your system runs. Automating deployment, this is actually fun for engineer to figure out how CI/CD works and then implement that.**

**As a technical person, I'm sure they will love that. I liked playing with GitHub actions recently – I didn't know how to use them. I spent like a weekend figuring this out and it was fun. So all these things that we talked about, they make sense and they are implementable. But why isn’t everyone actually implementing them? Why do we have all these problems that we talked about? Why do they still happen?**

44:12

Chris

That's a good question. I think there are a couple of reasons. One is, there's some people who are on the journey. They do those actions and they write a couple of unit tests against their Python code. And then they automate. Instead of manually deploying the steps, they use CI/CD, they have some unit tests, and then they still run into problems. So it's partly DevOps, they sort of have this model of, “Oh, I use Git. I'm doing DevOps. It's good.” But partly like, “How do you actually prove that things work? What's provable that a data system works?” And then “How do you optimize – how do you have the whole system, as opposed to just your part?” Those two things are hard for people.

So number one is like, “To prove data systems work, you have to pour data into them. You have to run them with data that's realistic.” Sometimes – in a startup, maybe – you're dealing with marketing data and you can just run the whole dataset over again. I'm a big fan of more functional data systems, where the raw data is immutable. You can run it from cloud resources that are cheap, disks as cheap. If you can run your whole system from start to finish every time. So I would say that a data engineer in development should have the raw data immutable and just run the whole thing – the Spark jobs, the visualizations – and make sure that you can prove it. Then have that be the new version of production. Other ways if you're deploying, that's the end-to-end the challenge, then there’s the definition of “What's a good test? What's a good data test?” This is a challenge. Getting good and accurate refresh test data that fits GDPR is a challenge.

Sometimes, actually doing the automation of environments – if you've got to GitHub Action, well, what proves that that development system works? Well, you've got to have test data. You've got to run your system end to end against that test data. That means you've got some hardware disk startup/shutdown – you're doing some DevOps-ey work. None of that stuff is not doable. I'm just saying it's work and it's really important work. You shouldn't just devote 1% of your time to it – you should think about 15% of your time on that. That may seem like a lot, but it's not. [laughs] Companies who do that, they end up doing more work – they actually get more done. You're stepping back to do more. Conceptually, I think there are still some gaps in what people think on the application of DevOps principles to data and analytic systems. I think there are still some gaps in optimizing the part versus the whole.

I think there's some gaps in environments, and I think there's just gaps in leadership, honestly. I just there's a lot of data and analytical leaders don't get it yet. That's sort of why I talk to people. I think it's still early. I used to run software teams in 1999 and I could ship software every three months, and I was hot shit. I was *good* in 1999. I ran a team of 30 people and we could ship software every three months. That was like “good startup best practice”. Right now, I would not get a job. [laughs] I would say, “I know how to ship code every three months!” People would go “What?!” And I'd say “You can't do it any faster!” And then people would go, “No, you should be able to press a button. When someone does a Git commit, the whole system should tell you if it's wrong.” And why? Well, people have learned what's “good” and what's “done”. It is a very different set of ideas. And that's what I want to have happen in data and analytics teams.

# Testing tools

48:25

Alexey

**Are there any tools you can recommend? You described all the difficulties that the teams face when implementing all these DataOps principles. It's not just about having CI/CD configured, but you also need to think about integration tests, the data has to be GDPR-compliant. How do we actually run these tests on data? Are there tools? There is also a question in the chat, for example, “Do we use things like Great Expectations? Or are there any other tools that we can use to make these steps easier?”**

48:59

Chris

Yeah, a lot of really… I think DBT has tests. Great Expectations – a test engine. You don't really need it. You can write tests in a lot of ways pretty simply. You can do row count checks. You can write these expectations and write SQL queries to do the tests. So the whole idea is that those tests themselves are done automatically, that they're in version control, that they live fairly close to the code that you're running on, and that they're run during production, and that all those tests are run during development. So, that's it. Think of – 10% of your work should be developing automated tests.

If you're a software engineer, and someone wrote 500 lines of software code – and there were no automated tests, you’ve got a problem. And tests are different. Like, unit tests are nice, but you also need system tests. You need to run the whole thing from end to end. When we first started the company, we had a small customer and I was the data engineer for it. I was sort of writing our product and doing data engineering. There's the inevitable problems, and I just write more tests, and our product helped us do all that. That's sort of why we're a product company – why we build a product – because we want to make those automation things easy for people.

# DataOps vs MLOps

50:42

Alexey

**Okay. There is another question. “What are the differences between DataOps and MLOps? Are they the same thing or different?”**

50:55

Chris

There are two answers to that. One is – as an engineer, no. It's the same freaking idea. It's just DevOps and Lean applied to data, so you call it DataOps or MLOps. Gartner’s been using this term XOps. So I don't really care – the principles matter, not the not the moniker. Right? [cross-talk]

51:15

Alexey

**When I say this, people roll their eyes and say, “No, MLOps is different, because it's about machine learning.”**

51:21

Chris

Yeah… you know, you're not different when you do machine learning. I mean, it's cool, right? But like, it's code that acts on data, just like ETL code, just like this code. You're building an application. You have more investigating – those two cycles are a little bit different. But yeah, I've gone to conferences and had data scientists basically telling me I'm wrong – that data science is different and that these process, DevOps principles don't apply. No, they don't apply directly. You need to adapt the concepts to models. And you can call that MLOps. You need to adapt them to data science and visualization.

I use the term DataOps to encompass… I think, the data, the model, the visualization, the governance – I think of those as a unit. So optimize the whole of that, not just your part. So do “Ops” on all of that. Just think of something really simple, like “I add a new table to a database.” Really simple. I want to add a new table to the database, so maybe I’ve got to change how it's joined to another table. Maybe that ends up as a new feature in the model. So the model’s gotta be retrained. And then I want to take the result of that and change my visualization. And then I've got to change my data catalog. The principles say, “Automate all of that.” Right? Put that all in version control. Deploy it all as a unit as once. The new code, the new model, the new visualization, and the new governance, the new data catalog – all those things are versional. They're all just code, right? Then, in development, pour some good data in, run tests against the new thing that you developed – the new table – and run all those other tests that you've had over the years to see if you've broken anything else. It's a very simple concept, but I think it does take some work to do that. And I think that's where people are struggling, because they need some automation to help them do it.

# The Head Chef at Data Kitchen

53:33

Alexey

**So your position – your title – is Head Chief at Data Kitchen.**

53:39

Chris

I’m Head Chef, yeah.

53:41

Alexey

**So can you tell us more about what you cook there?**

53:45

Chris

[laughs] As you know, I am a suburban American man, so my cooking is restricted to grilling outside – grilling meats outside. I grew up in Wisconsin, so one of my favorite things is bratwurst. With sauerkraut root and beer. In Milwaukee, Wisconsin, that's like the thing. I had Braunschweiger for lunch yesterday. So I'm like [laughs]

54:14

Alexey

**Pretty typical in Germany. [cross-talk] There’s a very large German population there, right?**

54:23

Chris

Yeah. In the central part of the US, there were a lot of Germans and Scandinavians in the late 1800s. Where I live now, in Boston, there's not. I mean, Germans themselves are sort of the largest ethnic group by count in the United States. Or maybe they were, I don't know. Maybe it's Hispanic now, I don't know. But, yeah. I'm a Head Chef just because it's fun. Our company, we're 45 people. We've been around for eight years. We're profitable. We're growing. And we're on this mission to get people to adopt DataOps and we want to sell some software along the way. Yeah, we do.

We think our software is great. But more than that, the reason I do this is I want people to get the idea and see that they don't have to live in this way. My feeling is that it's actually – I'm very optimistic – the fact that you're playing around with GitHub Actions, that's fantastic. I just think people are getting it. They're going to get there and it's going to become common practice five years from now. You know, the definition of “good” and “done” is just going to switch, and people are gonna go “Screw this, I am not working.” I remember, on StackOverflow, there was this – Joel's had this list of questions to ask your team.

55:47

Alexey

**12 questions, right?**

55:49

Chris

12 questions to ask, yeah. There's going to be that for data teams. And it's going to be *very* similar. There's gonna be DataOps in these questions.

55:56

Alexey

**You should put this together.**

55:59

Chris

[laughs] Yeah, is that. Yeah, “Ask these 12 questions and don't take a job.” [laughs] If they're like that. All those were (I remember most of them) like, “How much DevOps are you doing, really?” “How agile are you when it comes to DevOps?” I think the same answer is going to come. And I think there's more people who are going to stop and rebel against heroism and rebel against fear, and say, “There's got to be a better way.”

# What’s grilling at Data Kitchen?

56:32

Alexey

**Okay. So you said as the cook, you like to grill data at Data Kitchen? So what is the software you mentioned? What are you grilling?**

56:40

Chris

It's a platform to automate. We do some of those things that we talked about. We help people. I guess, the first thing is – we're the factory. We allow you to plug in your data science tool, your data engineering tool, your data visualization tool, your governance tool, and then we can observe all those in production, run tests against them, and we can help build environments for you. And then if you want, we could do even more. We could orchestrate all of them.

What we're trying to do is help people where they are and where they start. We tend to sell – we have a different way of building the company. We're profitable. We never took any financing. So, we're trying to grow a reasonable company and stay along for the long term. It's actually been good, because there's a lot of companies in the DataOps space who've been funded and growing, like, Superconductive just got an enormous amount of money. And a bunch of observability vendors that they put… [cross-talk]

57:50

Alexey

**Monte Carlo, right? They raised quite a lot of money.**

57:53

Chris

Yeah. So that's good, right? Because they'll be able to use that money to talk about these ideas. And that's a really good thing. People are going to hear it more and more and eventually it'll change.

58:06

Alexey

**Yeah. We had Barr on this podcast talking about that. It was a really good talk.**

58:09

Chris

Yeah, yeah. I think that's fantastic. My mindset is – I was so far out in the wilderness talking about these things six years ago, [chuckles] like “Is there anyone else talking about it?!” It warms my heart. [laughs] I think it's going to be, from a software standpoint, open source/closed source. I actually think from a career path, these DataOps engineers are a good career path, actually. Because I think people are going to need them. I think companies are starting projects and their people are getting these approaches.

For me, I guess, just technically, I did enough data and analytics that like, “Okay, I can do models. Maybe I'm not the world's best data scientist, but I can do some. I can do some visualization. I can do some ETL.” I can do it, and those are great, but my perspective is the system. You get the system right, you get iterations right, and errors right, and productivity right, and measurement right. You get those four things right. You're going to be able to power through.

60:17

Alexey

**We should be wrapping up, but there is one question. Do you need to go somewhere, or do you have a couple of minutes?**

60:25

Chris

I have a couple minutes.

# The DataOps Cookbook

60:27

Alexey

**I know that you have a book called “The DataOps Cookbook,” so I see that you love these cooking metaphors. Of course, what kind of books would you have in a Data Kitchen? So maybe you can briefly tell us what is in this book? I know this is a free book, so everyone can just go to your website and download it. Maybe you can briefly tell us about the table of contents and what people can learn from this book.**

60:52

Chris

Yeah. If you want to learn more, there's the DataOps Manifesto. That's 18 points, one page. There's a cookbook, which is about 150 pages with lots of pictures. And it goes one level deeper on what I talked about, like “How do you write a test?” “What is a good test?” “How do you organize a team?” “How do you track errors?” It's got lots of pictures and examples. Then we've even got a three hour video with questions, certification if you're wanting to do that. So we've invested a lot. We have a second book too, if you're more of a manager, and you're like, “These ideas are good. How do I get my team to adopt them? How do I lead DataOps from the top down?” So we’ve spent a lot of time making sure that people have these free resources to learn and to spread the ideas.

61:48

Alexey

**Yeah, thanks a lot for putting this together. These are invaluable resources, I will make sure to include all these links in the description so that everyone who is watching this can find them. And thank you for joining us today, for sharing your experience, telling us your stories. Thanks a lot for that. That's great. Also, thank you everyone for joining us today, for watching us, and for asking questions.**

62:14

Chris

Thank you for the opportunity. To all the listeners. It was good. Hopefully, you enjoyed the storytime and stayed awake.

62:24

Alexey

**[chuckles] Well, have a great weekend.**