1:20

Alexey

**This week, we'll talk about teaching data engineers. We have a special guest today, Jeff. Jeff has been teaching for quite some time. First, he was teaching data scientists, then data engineers. He will probably tell us more about that. So welcome, Jeff.**

1:34

Jeff

Hey, thanks for having me.

# Jeff’s background

1:36

Alexey

**Before we go into our main topic of teaching data engineers and teaching in general, let's start with your background. I know you have an amazing background, but it's probably better if you tell us about that. So can you tell us about your career journey so far?**

1:51

Jeff

Sure. I started as a lawyer and I went to law school. In my last semester of law school, I joined a tech startup. I was doing things like strategy for them – basically helping them expand their product and their offering. What I saw was that I started getting a lot of the questions that I wanted to answer and a lot of the work that I wanted to do just involved code. If we wanted to know what regions to move into, who our best salespeople were – we had to answer those questions with SQL and things like that. So that’s when I started coding.

And then I joined a law firm, because I had to pay off my student loans. But while I was there, I knew that I wanted to move into web development and learning how to code. I did General Assembly for three months. I was lucky enough to get hired by an awesome tech company. I really learned on the job for a couple years. After, I felt pretty good about being a developer and like I was really a contributor to the team for a little while.

I found myself actually reading a book on how to be a better teacher, which was weird because I never taught at that point before. I was like halfway through the book, and I just knew that education was a passion of mine. So I looked to see Flatiron School was hiring anyone and they were. They were looking for a new lead instructor. I applied on their website and started teaching there a few months later. I learned a ton there. I stayed there for about three and a half years, and then started my own school.

3:36

Alexey

**Yeah, that's pretty amazing – from law to software engineering, and then to teaching. That's amazing. So the book was How to Be a Better Teacher, right?**

# Getting feedback to become a better teacher

3:47

Jeff

No, the book I was reading was called Teach Like a Champion.

3:52

Alexey

**Teach Like a Champion. I should check it out. What does it talk about?**

3:56

Jeff

The main takeaway I got from it… It’s actually for like high school/middle school teachers. I volunteered in high school, so you just see how much time can be wasted because the classroom time is only like 45 minutes long. So, literally, if you spend like 10 minutes getting people settled and then handing out the assignment, then you have like 30 minutes to teach a lesson. It’s crazy.

But the main takeaway I got from that class was just feedback. Always try to be getting feedback from your students to see what they are actually learning and assess that constantly. So they would just give different mechanisms as to how to do that. And that's what I started working into the classroom and also what they did at Flatiron School.

4:46

Alexey

**Interesting, I should check it out. So “getting feedback” is the main takeaway, right?**

4:51

Jeff

Yeah, there's an article written by Malcolm Gladwell on teaching and educators. That's basically the way – because there's a saying “How do you evaluate teachers?” – and that's the main way that they evaluate teachers. And one thing I'd say is that you'd be surprised – you could do an awesome job teaching a subject, but that does not mean the students understand it.

I think teachers are often surprised about that, I often watch lectures given by these great Stanford professors and things like that, and then I'll see them give the quiz and I'm just surprised that, “Okay, the students did not retain this information,” because they're either doing other stuff, or there's so much to focus on or etc. But you're just always surprised when you go to assess student learning that it takes a few times for them for it to sink in.

5:40

Alexey

**It's always the case, right? No matter how good the teacher is.**

5:44

Jeff

Well, the other thing is – you know passive learning versus active learning? One thing from another book, they said “Look at who's making the noise in the classroom – if it's the teacher, he's the one having fun, he's the one learning. If it's the student, they're the one having fun, and they're the ones learning.” So you can tell where the learning is going on just by who's the group of people being active.

# Going from engineering to teaching

6:13

Alexey

**So you were a software engineer – you were doing development work – and then you accidentally bumped into this book and decided to become an instructor? Or how did this happen? Why did you decide to actually leave your software engineering job and become a teacher?**

6:32

Jeff

I have been interested in education since I was about 20 years old. I did a lot of human rights stuff in undergrad. So I did this really cool summer program working with refugees and got really into it when I was like 20. I wanted to kind of keep going with that in college. So I was just looking for projects to help out similar types of groups and one thing I kept on hearing from people is, “Hey, can we donate this stuff to them? Or can we do this type of thing?” And they're like, “Alright. Well, they need cleaner water. So okay, you can give them water filters. But unless there's training to go along with it, they're not really going to know how to use it.” And they said, “By the way, instead of water filters – they could just boil their water.

If you train them to do these sanitary things, then they actually don't need these services in the first place.” So we kept on running into education as a solution. For me, it really came from that element. Just reading about it – it involves so many different skill sets, it was just a great challenge. You could go in a lot of different directions with it and go very deep with it as well.

7:50

Alexey

**It's like, instead of bringing people fish – teaching them how to fish, right?**

7:55

Jeff

Yeah, but it's also like the last mile problem – unless the technology is really perfect. Okay, well, you hand it over to the people, are they actually using it? Also, when I went there, they were basically asking for education. They needed work. They’re refugees and they don't speak English. So it's like English training. In college, I was trying to develop an online program to help them learn English and things like that. So I just kept on running into education as something that seemed like a good solution, and it also was fun to do.

# Jeff on becoming a curriculum writer

8:35

Alexey

**So you were studying at Flatiron or…? You said you studied at General Assembly. So this was there?**

8:42

Jeff

When I was a lawyer, at that point, I really wanted to learn how to code. I just worked in a startup. I saw 10 people doing this amazing stuff. It blew me away. So when I was a lawyer, I was just itching to try to figure out how to code. At that point (it was 2013) there were only a few bootcamps for months that I could see. Right now they're called Recurse Center, I think, but at the time they had a different name. Anyway, they were more like a writers’ retreat for coders – for people that really could code. And then Flatiron School and General Assembly were really the first two coding bootcamps in New York City that you could get into. I didn't get into Flatiron School – I think they were filled up – so I went to General Assembly.

9:36

Alexey

**Okay. You were looking for a teaching job, so you went to the Flatiron website (the careers page) and you saw a vacancy for a lead instructor. Then you worked there as a lead teacher. And I think your last position there, if I remember correctly, is Lead Curriculum Writer. So what does it mean to be a curriculum writer?**

9:58

Jeff

Essentially, my last role was really to build the data science course. Because at that point, they offered the web development course and they’d just been bought by WeWork, and they wanted to develop – both expand the school in terms of locations and also their product offerings. And I was really interested in data science. I kept on asking, even before that, if we could build a course on that. It basically involves first pitching to leadership that, “Hey, this would work, both in terms of employers hiring these students and the fact that students would want to apply for that.”

There's a lot of competitor analysis, looking at the job market, looking at whether there is a viable career path, looking at student interests, things like that. And then it was also looking at other schools to see how they developed a curriculum and what subjects they taught, what it would mean for us to do something like that, talking to employers to get feedback from them, talking to past students from these other schools, and then just getting started writing curriculum. Then about six months in, we hired a team of curriculum writers to help out.

11:15

Alexey

**So this is what you did as a lead curriculum writer. Right?**

11:19

Jeff

I guess so. I mean…

11:22

Alexey

**This wasn't your title. As you said, you were web development, right?**

11:27

Jeff

Right. I mean, my title didn't really… I don't know. I also then taught the first course while I was still the lead curriculum writer.

11:35

Alexey

**Okay, but what does it actually mean to be lead curriculum writer?**

11:38

Jeff

To be a lead curriculum writer? It means you write curriculum.

# Creating a curriculum that reinforces learning

11:44

Alexey

**Yes, but what does it mean to write the curriculum? What does it mean to *build* the curriculum? What do you do there? What is a curriculum, actually?**

11:52

Jeff

Oh, okay. First, you basically write out the syllabus and you try it. First, I guess I read and just got a bunch of ideas by looking at other people's syllabuses and topics and reading blog posts on the school experience and things like that. Then from there, you start to collect [cross-talk]

12:13

Alexey

**Just general level topics, right? It could be regression, classification, clustering, time series, whatever.**

12:19

Jeff

Exactly. Right. And then looking at Coursera courses, looking at statistics courses. You know, you kind of start with, “Okay, here are the topics we need to teach.” And then “What does that mean?” You go deep into these individual courses that are all kind of disparate and then you start to see how they line up. You always want the learning material to fold into one another. If you just teach it once and it's never used again, your brain learns that and you forget it immediately. So you want to see, “Okay, how do these topics build on one another?”

Then you want to develop a syllabus where that lines up. Finally, you start writing the curriculum. You just start literally typing on the page, coding out and so on. I started with the intro material and just started writing that curriculum. As you're doing it, things change. Especially as you then go to teach the material, things change as well. The ordering changes, or you need more time here, and not as much time there, and things like that.

13:25

Alexey

**The curriculum is a detailed description of each unit, or? [cross-talk]**

13:31

Jeff

I mean, it's a syllabus, but then it's also literally readings. So you have to write out lessons for every corresponding lecture that you give. So each lecture – say the lecture is an hour – that probably consists of two to three different lessons and then probably two to three different labs. So you'll maybe give two different small labs, and then one larger lab to tie it all together.

14:01

Alexey

**I see. So it could be like a step-by-step instruction of what exactly you're going to talk about. For example, if we take regression – first you show a data set, then you talk about importing SciKit Learn, then you talk about turning this dataset into something. Then you said there is a lab, so students actually sit and do that. And then there is another bunch of sessions of you talking about something, then students do something. So you describe all that on a piece of paper or in a Google document.**

14:30

Jeff

Yeah, but I think you break it down very granularly because you want to understand each component of y= mx+b (or whatever). I think with regression it would be you would start with, “Okay. Well, this is B” Because that's the most simple probably to understand “and it just raises the line up and down. And then M, of course, is the slope.” And then from there, you’d probably be like, “Hey, by the way, we can build this and plot this out in code. As we change these numbers, as we change M and B, you can see this thing change.” So you want them to understand each component conceptually and then from there, you probably want them to really understand the application of it. Ideally, you want to understand the application first, and then go into the underlying components.

15:20

Alexey

**Did you already know about data engineering back then when you wanted to teach it? [cross-talk]**

15:24

Jeff

No, no, I had no idea. When I was teaching data science, I'd say one thing I saw was – because I had been teaching web development at that point for a couple of years – one thing I kept seeing, and kind of kept on saying to myself and to some others was like, “Man, if they knew they're better at engineering, they would just be so much better at these projects and the skills.” You could just see how much it made sense.

The reason why I was able to teach myself SK learn and PyTorch and things like that pretty quickly was because I was a software developer. I could read libraries pretty well and understand what was going on. I kept seeing that and kept trying to advocate for more and more backend development work. When I first launched my school, it was data science as well. And really, the turning point was just that the job market shifted, so that I felt like it was no longer a viable career path for someone to just go through a coding bootcamp and become a data scientist. But it was a viable career path to become a data engineer.

# Jeff on starting his own data engineering bootcamp

16:37

Alexey

**Maybe, let's go a little bit back. At some point, you decided to leave Flatiron and start your own bootcamp. Why did you make this decision? Couldn’t you just follow the same approach? You could have pitched this to the school, right? [cross-talk]**

16:58

Jeff

Well, I didn't know that data engineering… I left Flatiron School before that. But when I first taught at Flatiron school – I think the first course graduated in 2018 and 2019 – students got jobs as data scientists pretty quickly, even though the course still had a lot of improvement. But the job market was such that that was okay. They were getting great career paths out of it. So that was really cool. In terms of leaving Flatiron School, I guess there are other components that I wanted to do beyond just building curriculum.

There were other things and problems that I wanted to solve. One thing was career services. I felt like there was a huge opportunity in career services to just stick with the students. A lot of times what I saw was that students would be at this level – maybe right at the bar to get a job when they graduate – and then there was a huge difference between students that just had a good path going forward and those that are floundering. I would see it because we would check in on students that didn't get a job three to four months later it was like, “Oh, crap. They forgot so much of what we had been teaching them.” So that was one thing.

I thought the school could be way cheaper, by making it part time, by just lowering the tuition, things like that. I think like, for me, I also had this question of, “Okay, why doesn't everyone do this?” And I didn't feel like anyone had a good answer for that. I think when I left the school, it was things like that too, “I want to answer those kinds of questions that don't have to do with curriculum. And right now, I'm a teacher and I write curriculum.” So to do that, this is now starting to be my own school.

18:56

Alexey

**So to lower the price, you wanted to do it part-time for those who already studied there. [cross-talk]**

19:04

Jeff

That was really a dramatic thing. That was part of the “why doesn't everybody do this” type component. When you join a boot camp it’s always “Take the leap and trust”. And it's a boot camp, which means it's an all-encompassing type of thing. And I wanted it to be something where you don't have to put your whole career on the line to do this thing. It doesn't have to feel like… one, I went through boot camp and I know that it feels very vulnerable when you quit your job, and then you just put in so much trust in a school to really deliver. You have no other option.

At that point, you are in their hands. And if you don't quit your job, well then all of a sudden, that's not true at all. You know what I mean? If the school doesn't work out, or you don't like it, you can just walk away, effectively, and it's not a huge deal. But if something happens – you get sick, there's some sort of tragedy, or whatever – yeah, that's pretty dramatic when you quit your job and you're trying to transform your career in four months.

20:08

Alexey

**You saw that and you thought, “Okay, I should create my own school for people who will do it part time,” right? [cross-talk]**

20:18

Jeff

It was also “lower the barrier to entry.” Like, how do you lower the barrier to entry to make this possible? Really the first question was, again, “Why don't more people do this?” One thing I started doing was just teaching weekly workshops – right after work. Because that's a natural step to just showing people they can do it. It's not a huge commitment. Then, instead of showing up once a week, show up three times.

Obviously, there’s some more commitment but hopefully, you see that, “Hey, coding is more fun than you might think. It does not involve math and it's different than you might think.” The misconceptions around coding, from people who have never coded before, it's pretty dramatic. They were for me. So it was a lot of that, “How do you introduce this to people? How do you not make it such a huge step and make it an easier transition for people?”

# Shifting from teaching ML and data science to teaching data engineering

21:17

Alexey

**Yeah, interesting. You started first with machine learning for your own bootcamp – machine learning, data science, all that – but then you gradually shifted to data engineering. Why did you do that?**

21:28

Jeff

Yeah, it was all about the job opportunities. I was very surprised. Like I said, when I first started teaching data science, we went on people's LinkedIn who went to different boot camps and made sure there was a real career path. Same thing with the students that I taught, initially, they made the leap and they got to become data scientists. Some of them are doing some really amazing things now. My first data science class that I was teaching, I started talking to employers and telling them what my students are doing and things like that, and they replied, “Well, we still wouldn't be that interested. They still need more. They're still not actually great candidates. Do you know how many applications we get for this position? It’s crazy. I don't even know why I'm talking to you.”

22:21

Alexey

**How many? Do you remember?**

22:23

Jeff

Well, I remember… You could click on LinkedIn pages – so many of the jobs will say over 500. I spoke to a data scientist at BCG, who told me, “I get 20 pings on LinkedIn a day asking me ‘how do I become a data scientist?’ The only reason I'm talking to you is because you seemed interesting and you have this school.” You can read Vicki Boykis’s blog, where she wrote that blog about “data science is different now,” and she talks about just so many people.

One of the things I wanted to point out is just, I would be like, “Yeah, but my students are really good.” And they were like. But it doesn't matter if there's just so much noise flooding the market. It's just not worth the time. I think the reason why people are looking for Master’s and PhDs – it’s just an easy way to cut out a bit of the applicant pool, just for mental sanity purposes. And then, you’re like “Alright, I'm just going to look at these people.”

23:26

Alexey

**And data engineering, on the other hand, was in demand, but did not have a lot of attention like data science. Right?**

23:35

Jeff

Exactly right. Yeah. One, you actually had engineering skills. If you graduate with working in data engineering – with data science, there's so much to learn, too. That was the other thing. It was hard to build a curriculum around it. I kept extending the course. It was originally six months, and then was like, “Okay. Well, we should teach AWS, Docker, and Airflow as well.” So I kept on extending it to the point that it was like, eight-nine months.

With data engineering it was a more defined skill set of Python, SQL, Cloud computing, orchestration, things like that. You could go deep into those subjects, so that you weren't just going an inch deep in 15 different subjects – you're focusing on really giving them solid Python SQL skills which are turning them into back end developers. Then also you add this data specialization on top.

24:32

Alexey

**Did you also see that some of the students who went through your data science bootcamp were getting jobs as data engineers, or not really?**

24:42

Jeff

No, we were able to get them… there were qualified students, like PhD students and people that had been working with SQL or things like that for five plus years. It was successful, but I didn't think it was sustainable. I thought it was still too much lift to get them there. I thought, “This was way too much work.” They crushed Kaggle competitions before they even graduated. They did so much stuff and I just was surprised with the resistance. But I did talk to graduates from other bootcamps because I was wondering, “Are they able to get data science jobs? How does this work?” And when I called the students and spoke to them, either they would get engineering jobs or data engineering jobs or analyst positions. But it felt like that door had closed.

25:39

Alexey

**Interesting. I talked to a data science boot camp here in Berlin and they said something similar – that many students that graduate from data science bootcamp end up being hired as data analysts. Therefore, they now are repurposing this, or rethinking their strategy – building a new curriculum for data analysts – because of how data scientists are usually hired. Meaning the data scientists that graduate from that bootcamp.**

26:09

Jeff

Yeah. I even wonder how much… I trust a person I’m hiring for machine learning more than data science to begin with. With data science I feel like there's a big diversity in the skill set asked. With machine learning, I think it's becoming more consolidated as a skill set being asked for – it's just pretty advanced. I think you need the data engineering skill set plus the machine learning skill set.

# Making sure that students get hired

26:40

Alexey

**You have students who go through your bootcamp for data engineers, so how do you make sure they get hired?**

26:50

Jeff

We do a lot. One crucial thing is, honestly, the admissions – only admitting people that we believe are going to get hired at the end of it. That's the main question I asked myself. It's hard to do, because one, you like the applicants and you want to say ‘yes’ to people. Then, obviously I want to grow the school, but only doing that – I think that's a huge thing. Two, is the curriculum, we just try to have it line up perfectly with what employers are looking for. And then three, we do post-career work – we meet with them twice weekly to make sure that they're on track.

That also gives us feedback on the curriculum because I see what questions are being answered, I see how they're applying for jobs, things like that. Then finally, the only thing we saw on the first day of the data engineering course – we paired students up with employers after they graduated, working for free, because this way if it takes a few months to find a job, they're building experience. That was really successful, so we built that into this program – halfway through, you'll start working for a company for free. That way you have experience by the time you graduate.

28:07

Alexey

**Interesting, how does that actually work? Do companies often agree to this? Sometimes some random people write to me on LinkedIn, “Hey, I will work for free. Give me a job.” And then I'm like [chuckles] “Okay, but how do you do that?” And then another thing is, if a person is working for free, I do not trust the motivation of this person. They might just decide one day not to show up, because why would they? So I have very mixed feelings. Another thing here is like, “Why for free?” Can’t they just pay minimum wage for that as well?**

28:45

Jeff

Yeah. So there are a couple things that we saw too. If students just asked for an internship, it's still an investment on the side of the company. The most expensive thing is still going to be a senior engineer’s time to make sure and do project management and things like that. So that's why we kind of said, “Okay, we'll help you with coaching the students and kind of be the manager for the students.” So it's like they deliver a good amount of stuff for us.

The other thing is, we basically say, “Hey, find projects that are not mission critical, but are nice to have and will really provide value. And if they're delivered, you'll use them.” Then, we also allocate in-class time. We allocate six hours per week and make sure students tell us in advance if they're available for additional hours, and that they stick to that. As a teacher, we provide a lot of, “Hey, this is what you signed up for. We have to deliver. We have to be engaged in it.” The admissions really helps – our students are professionals and they're quite good. They provide that professionalism when they do the job.

# Screening bootcamp applicants

30:04

Alexey

**When you do this screening, what kind of signals do you look for?**

30:11

Jeff

Technical skills we look for are just like… [cross-talk]

30:18

Alexey

**Do you need to be able to program?**

30:21

Jeff

Well, yeah. Yeah. [cross-talk]

30:20

Alexey

**[cross-talk] for engineers, right?**

30:24

Jeff

Say that again?

30:25

Alexey

**For people who are already engineers – let's say somebody's working in software engineering already. Is this for them, or they are not [cross-talk]**

30:32

Jeff

No. I mean, we have taken people like that. I would say some of our students – maybe about a third to 20% have had previous engineering experience or have gone to other boot camps and things like that. By the way, we've had people that have been CTOs for different organizations, but that doesn't necessarily validate their coding background or skill set. So we always do a technical interview. What we do is give them free curriculum that we have on the website, like “Intro to Coding” – starting from zero to whatever in Python. We say, “Hey, go do the first 10 lessons and then we'll give you an assessment.” When we're doing the assessment, what I do is look to see if they understand each step. Like they're not memorizing anything, but they can tell me why they're using each step. And then if I give them something a little bit off the track of what they learned in the curriculum, can they respond to my teaching style and understand this?

So you're looking to see, “Are they thinking?” It’s hard to do when you're under pressure and then translating those thoughts into code. If they can do that, that's pretty good, right? If they've only spent a couple of weeks and they were able to do that – that's pretty good. You also pick up a lot of things like, “Alright, are they going to put in the work? Are they motivated to put in the work?” Those components. I've already spoken to them at this point. In the earlier stage, we see “Okay, what are they looking to get out of this program? What do they know about the industry? What's their previous background that employers might look for and find attractive when they then go to hire them?” So there are things like that that we're also looking at.

# Knowing when it’s time to apply for jobs

32:31

Alexey

**I have a question that is quite relevant to what we're talking about, “How does one know that the students are ready to apply for an entry-level data engineer role?” Maybe I will rephrase it a little bit, because it's similar to a question I wanted to ask, which is “What do you include in the program?” I think they are similar because you probably know what entry level data engineers should know. You put these things specifically in the curriculum, right? Am I correct that this is how you know that they are ready? Because you include only the things that they need to know? [cross-talk]**

33:05

Jeff

Yeah, we can put it in the curriculum, but it goes back to the same thing – they need to master and really understand that material. Not only that, it’s really 400+ hours of material over six months. Trust me, when they graduate, to a lot of them, I'll say “Apply. Start applying for jobs, even if you're not ready. Alright? Start applying. It’s cool. Get those rejections out of the way.” And that's a good motivating factor and we start to see. So we do that.

The other thing is, we'll give them technical interview questions and see how they perform in that. But there's no harm in applying to the job, especially when they're starting to get takers and people that accept them, which you generally get when you have those skill sets. That's a nice thing – they don't have too many problems getting interviews. So I encourage them to get the interviews. Get scared. Bomb an interview. Then we can use that and they'll be motivated to improve.

34:13

Alexey

**Basically, what you're saying is – they might not be ready, but they just need to get over this fear of rejection.**

34:22

Jeff

I mean, interviewing is a skill in and of itself. You see all the time on LinkedIn, how the technical interview does not line up to the job. Of course, we all know that. Being great at interviewing is itself a skill, probably no matter what job you're applying for. So I want them to start doing that and it will start to put them on track as well as – they bombed the interview, now they really want to start practicing LeetCode. Now they see like, “Oh crap, I really need to improve. I thought it was good at SQL, but now I’ve really got to improve it even more.”

35:01

Alexey

**Okay. Technically, from what I understood from you, the students are already ready technically, but they just need to learn how to pass the interview – what kind of questions get asked, how to answer these questions. And these questions are not always technical. From my experience, maybe 50% of the interview is not super technical. They're like, “Okay, tell me about yourself. Tell me about the project that you're proud of. Tell me about x, y, z.” And you need to have some practice in answering these questions. The only way you can learn is I guess going into interviews. Right?**

35:35

Jeff

Well, that's true. But I think it really helps to talk to an engineer about your experience – like if you have a friend that's an engineer. What I'll do is, before students go on interviews, I have every student send me their resume. I look through their resume and then I talk to them for like, 30-40 minutes about their job experience. I'd say inevitably, I'll be like, “Oh, I didn't get any of that from your resume. That'd be super attractive to an employer. Let's put that in there.” That process basically entails that what we put on the resume now focuses on what you want to talk about in the interview.

# The curriculum of JigsawLabs.io

36:18

Alexey

**Okay. What do you actually put on the curriculum? What are the topics there?**

36:22

Jeff

The first section, we think of it like analytics engineering – so it's Python, with a strong focus on SQL. Then it’s building an analytics engineering pipeline – so Fivetran, DBT, Snowflake. Then we use Mode as a business intelligence tool. So that's the first section.

Then the second section is essentially backend engineering. So it's Flask, building ORMs, the adapter pattern, ETL in Python – MVC, obviously. And that takes a while to go through that and testing. A lot of that is like, “How do you write code for a larger codebase? How do you navigate a larger codebase?” Things like that. That's another 10 weeks or so.

Then, finally, we go into cloud computing and Airflow to Docker, AWS, Airflow. We also layer in, starting in that second semester, one – they start their internship (we do that for six hours per week) and then also, we start layering in interview questions. So they start kind of thinking that way.

# The market demand of Spark, Kafka, and Kubernetes (or lack thereof)

37:41

Alexey

**I'm making some comparisons and parallels with the course we have – Data Engineering Zoomcamp. We have this data analytics engineering module, but we don't really talk about Fivetran, we cover just DBT. But we do cover things like Spark and Kafka, which you do not cover, right?**

38:05

Jeff

We do not cover that. We taught Spark in the first iteration, so we have curriculum on that. But we saw that it wasn't really asked in interviews and it wasn't really required of juniors. I see it a lot for our senior engineering positions. We scraped all these data engineering positions – it does come up a lot. I wanted to look deeper to see if it's even listed for junior data engineering positions – like zero to three years experience – and it wasn't asked in interviews. So that's why we were like, “Okay, we have other things to teach.”

38:43

Alexey

**Okay. And Kafka I guess. I remember when we first met, it was a couple of years ago and I think you were showing me some curriculum and you were asking for my feedback on it. I think we talked about Kafka at some point, right? You wanted to include this, but at the end, I guess you decided to drop it. Was it the same thing as with Spark?**

39:03

Jeff

Yeah, it was the same thing. It's listed, again, for more senior level positions. The other thing is, this analytics engineering role really has grown in the past couple of years. So we wanted to focus and also put time into that. But the truth is, we started taking out things. We used to teach Kubernetes too. We saw that while it turned heads and people were impressed that the students knew it, and sometimes they would even ask questions on it, it took so long – it took like two and a half weeks to teach it, which is a good chunk of the course. But then it just wasn't enough value added to justify keeping it.

39:44

Alexey

**Yeah. Again, I remember our discussion a couple of years ago where you said that there is no good book that covers machine learning and Kubernetes and I thought, “Okay, I need to include this in my book.”**

39:55

Jeff

Right, right. Exactly. [chuckles]

39:56

Alexey

**It's a very complex topic. I realized that it can be very overwhelming for people entering the field.**

40:04

Jeff

You can teach it, I really think. You teach Docker first, obviously. We were able to successfully teach Kubernetes. The other part of it that was not great was that people stopped coding. You know what I mean? With Kubernetes, you're writing YAML files – you're not coding. And that's two and a half weeks that the students are not coding. That was the other thing – we always wanted reinforcing skill sets. So if you look at the curriculum – I tell people that are applying, “Yeah, the curriculum is really 85% Python and SQL.” That's really what we're teaching. All the time.

# Advice for data analysts that want to move into data engineering

40:42

Alexey

**A question we have is “What steps should a data analytics or BI professional take to become a data engineer?” We kind of partly covered that when you were describing the curriculum. Maybe they should start with analytics engineering. Right? What do you think about that?**

40:58

Jeff

For analytics engineering, or data engineering?

41:01

Alexey

**For data analysts – they want to become data engineers. So what kind of steps should they take?**

41:07

Jeff

Yeah, I see. The main thing to ramp up on is backend engineering and cloud computing. Cloud computing is probably the easier step to fill in the gaps with, and then Python. If you want to just start applying for jobs, maybe you start with cloud computing. But probably, on the job, a lot of what will be asked of you will be Python. And I think you're seeing more and more people asking Python questions in the interviews.

41:38

Alexey

**What about things like Fivetran, DBT?**

41:41

Jeff

Yeah, definitely. I mean, they're easier to learn. Fivetran you can learn in a day or less than that. It's designed to be very easy to learn. DBT – same type of thing. You can navigate DBT well enough for interviews probably in a week or two. It's not so bad. What's harder for ‘on the job’ is the DBT patterns – and you should know that stuff, like staging and integration and marks. So learning that stuff is valuable, but I think if you can just start navigating DBT and then more writing CTEs and writing modular SQL code, that will be helpful.

42:26

Alexey

**What is a CTE?**

42:28

Jeff

A CTE? A common table expression. So just wrapping the SQL statement or with a clause.

42:37

Alexey

**You said that you scraped a lot of job descriptions and you saw that Spark is not there for junior positions, but it's present for seniors. I’m wondering, how often do you see Fivetran and things like Airbyte – these lower-code things?**

42:56

Jeff

I didn't scrape specifically for analytics engineers, yet.

43:02

Alexey

**But for data engineers?**

# The market demand of ETL/ELT and DBT (or lack thereof)

43:03

Jeff

Well, yeah – I just did it for data engineers, literally in the past month. In the analytics engineering stack, it does not show up at all. It shows up, but I literally see like 10 out of 400 job descriptions or listings with DBT – which is kind of crazy. Even the words – ETL shows up at the very top, but then ELT is nowhere to be found.

43:27

Alexey

**I think people just use them interchangeably. Even I always confuse them, like “Oh, what is the difference? I'll just go with ETL. It doesn't matter. I just mean ‘data pipeline.’”**

43:38

Jeff

Yeah, but when you look at the Slack channels then you'll see DBT and Airbyte and things like that listed – in any of these Slack channels. But I don't know, in the job descriptions, I'm not sure if it's companies that aren't as plugged in or if it's just the job description isn't lining up or whatever.

43:57

Alexey

**But you still decided to teach it right? Regardless of whether it’s listed or not. [cross-talk]**

44:01

Jeff

[cross-talk] It depends on if there's a market for it. I know that we can go to the Slack channels or these employers and they'd be attracted to people that really know DBT. And the same thing when I talk to employers who are like, “Oh, do you have people that don't really know DBT? We'd love that?” and I'm like, “Okay, we'll make them learn it.”

# The importance of Python, SQL, and data modeling for data engineering roles

44:21

Alexey

**Another question we have is from John, who asks, “For data engineering SQL is a useful skill. How does one improve SQL if somebody wants to do that?”**

44:33

Jeff

Yeah. You basically need to know SQL in and out. Beyond aggregates, obviously, and joins – that’s kind of level one. Then you probably want to know window functions, which is a favorite of interview questions. You know, if you look at data engineering interviews, people will be like “Oh I always ask this LeetCode question. And I always ask this LeetCode question.” So those LeetCode questions – you should probably be at the medium level.

45:07

Alexey

**By “LeetCode questions,” you mean…? I know on LeetCode, you have algorithmic challenges and SQL problems as well. You mean the SQL ones?**

45:14

Jeff

I mean the SQL – just click on LeetCode for SQL and then be able to do probably up to medium – for most positions that’s pretty good. That's really the main way they'll be assessing SQL. The other thing is going to be data modeling, like knowing the difference in modeling between, OLTP versus OLAP. So practice modeling – you could probably find them online as well. That's definitely fair game for people to ask.

45:45

Alexey

**Do you know any useful resources for that?**

45:49

Jeff

OLTP versus OLAP?

45:51

Alexey

**Yeah and for data modeling.**

45:52

Jeff

For data modeling? You know what's interesting? Someone told me – another teacher that taught at Turing school, I think (or something like that) – they said, REST is really good for teaching data modeling, because it's pretty similar principles. So there's like REST for mere mortals. And then, of course, the Kindle book. But sometimes it's a bit too much.

46:17

Alexey

**It's quite formal, right?**

46:19

Jeff

Yeah, it's very wordy. I feel like the first couple chapters are enough of it. I think that there is some stuff online where you can do it. The other thing are those classic databases, like Microsoft has – like the Northwind database and things – those types of databases and see how they're modeled. Try to model them in advance. You can take any kind of domain – like an airport, right? I give my students that, “Okay, model an airport. Here's a ticket for a flight at an airport, you do the modeling for it.” Then you could show it to an engineer, or maybe if you google it online, maybe people already have an answer for it.

47:09

Alexey

**Okay, yeah. [chuckles] So just think of something like… I don’t know – a parking lot.**

47:14

Jeff

Yeah, like look at sample databases. Also, maybe if you go to code bases that are online. For instance, some organizations, if you look at their open source repositories – just go to their models afterwards and look at it. Draw it out. Even just drawing it out would probably be useful. But if beforehand, you guess “What is the relationship?” Or “How is this modeled?” That would probably be really useful.

# Interview expectations

47:46

Alexey

**So somebody with the nickname “futureDSengineer” asks, “When do you stop learning and start attending interviews? It seems like there’s an ocean of topics and it never ends. It overwhelms sometimes.”**

48:00

Jeff

Yeah. So that's why I think there's no harm in just attending an interview just do it – just for the experience. Maybe we can give some expectation as to what probably is going to be asked in an interview. The first interview would probably be a screening interview. Maybe they'll ask you a little bit about data engineering, maybe they'll ask you OLTP versus OLAP. Maybe they'll just ask you, “Have you built any data pipelines?” And, “Tell me about some of the tools you've used in data engineering.” Things like that. You don't need to know *every* one of them. But they just want to know that you've spent some time with this.

Then the second interview is probably going to be a SQL question. You can expect that, I'd say, if you can do some of the medium LeetCode problems, then you should feel pretty good. And you don't have to get them 100% right. But if you don't feel blown away by the medium SQL questions, then you're probably pretty good. At that point, go and interview. It's fine. There will be more interviews that show up. So I'd say, go to the interview and that will help you a little bit to self-assess afterwards. You can see where you are.

One thing I would say – one caveat to that and one thing to be careful of – sometimes I'll see students go on an interview and they'll BS something out of left field, and of course, they bomb it. Then they’ll be like, “I gotta learn that thing. I gotta learn everything about that thing.” Then you come back to them two months later, and they're still just learning everything about that thing. So you want to stay on your learning path. “Okay, I'm building a data pipeline. I'm improving my SQL skills. I'm going through some Python LeetCode problems.” Because those will also be asked.

# How to get started in teaching

49:52

Alexey

**Okay. There are still a couple of questions I want to ask. The question I want to ask most is “If somebody wants to start teaching software engineering or data engineering or data science – any topic – what would you suggest to them?” Let's say I am a data engineer already. I do not have experience in teaching. What should I do?**

50:15

Jeff

Okay. The first thing is – think of a topic, I guess, that is beginner level. Meaning that one, it’s something that a beginner can accomplish and two, would be interesting to a beginner. That itself will be a process. It took me multiple tries to get there. And then I’d say to explain it to someone. Explain it to someone at the level that you want your audience to be – just one person – and walk them through it. Teach them it and then see if they know it. Then also maybe hear their feedback on it. That would be pretty good, actually.

At that point, you're probably ready to deliver that to a meet up – to a small classroom – something like that. If you can do it in person, it's better because – one, it’s easier in person, actually, than online. And then two, you get more feedback. The feedback is almost immediate – you can sense everything when you're teaching in person. You can see their faces and things like that, which sometimes online, you can’t. So if you can just give a workshop in person – that will be great. And then do it again. Just give that same workshop to a different meetup. And revise it. That's kind of the teaching process –then revise the workshop, and think about it, and change the order of stuff, and do it again.

51:44

Alexey

**How do you revise it? Is it based on the questions you get? You explain something and then you see that clearly everybody is lost? Right? And then you say, “Okay, probably I should explain something else first.”**

51:56

Jeff

Sometimes it’s the questions – questions can be really good and give you a sense that people think what you’re teaching is kind of cool. Lots of times, you want to get to the point as fast as possible, or show the benefit as fast as possible. One of the reasons why I said teaching online is harder, is because online, people can just leave in a millisecond. If you're boring for like five minutes – they're going on Netflix, immediately.

If you're not interesting within the first five minutes, [chuckles] it's amazing to just see the people drop out of the Zoom. I found that one of my best lectures in person was building a neural network from scratch. Lots of people showed up and they got a lot out of it. But then I delivered it online, and because there was such a lead-up until we got to the interesting stuff in, over half the people left by the time we got there. So I just had to totally flip the order of everything to make it work. It told me a lot about “Oh, if I can actually put this here and then we can start getting to the point way earlier.”

53:05

Alexey

**Okay. When people come to the classroom, it's not so easy for them to leave and switch to Netflix because they are physically in that room.**

53:14

Jeff

Exactly right. They're physically there. They're with people. Things like that. You can also give them activities to do. In zoom sessions, it’s a lot harder. Once you tell people to do something, I found that's another point where people would just drop off. But in person, it's great. You're just like, “Okay, here's this activity. Turn to the person next to you and work on it with them.” That stuff is great.

53:41

Alexey

**Then you said “pick something beginner level and explain it to somebody.” How do I pick up this somebody? How do I select who to talk to?**

53:49

Jeff

Yeah, I tell my mom. You know what I mean? Really, if you go on my website, it’s like “Me teaching my mom how to code in 10 Lessons.” [laughs] [cross-talk]

54:00

Alexey

**You actually recorded that?**

54:01

Jeff

What’s that? [Alexey repeats] Yeah, I recorded it. Yes, it's on the website. [laughs] And people were like, “I'm signing up because I saw you teach your mom to code.” [laughs] She gave me great feedback, because this is someone who, one, has no background in coding, let alone email. And, two has zero interest in code. She's retired – she's not going to use this. When I'm giving these lessons, going through lists and “Here's how you select from…” you know, all that – she's like, “Why do I need to know that?” Which is how you're supposed to teach – everything should be directly solving the problem, so they want to learn it and you're answering their question. That's like how you see that that's a good lesson.

# The challenges of being a one-person company

54:54

Alexey

**All this time when you talk about the bootcamp, you say “we” “we teach” but I think you're a one-person company? Right? [chuckles]**

55:04

Jeff

More or less. No, I have a co-founder. But yeah, in terms of the education and things like that – yeah, it's basically me.

55:11

Alexey

**How difficult is it for you to run this? What are the main challenges?**

55:18

Jeff

You know, writing the curriculum was a massive lift – probably over 2500 pages of curriculum in the past three years. That's where my law background really helps – I can just write 8-page essays like they're nothing. Also the education.

55:39

Alexey

**[chuckles] That’s useful.**

55:40

Jeff

Yeah, I found a use for it. Thank God. But I'm still just writing constantly – really all the time, for such a long time. But I'd say that's kind of another challenge, which hopefully I did properly. When people asked me what I did, if someone just randomly met me, I’d say, “I'm a curriculum writer.” That's all I did for basically two and a half years. It was writing curriculum and then teaching the course 17 hours a week.

One of the nice things about this business is that I've been able to focus on one thing at a time. Before I started the course, it was marketing and giving these workshops and things like that. Then once the course starts, for the next six-eight months, I'm just writing curriculum and doing that. Then when students graduate, like right now, I'm focusing on getting them jobs and marketing the next class. That's it. I'm not writing curriculum right now – I'm doing those two things. So at least just having a couple of things on my plate and not five different things on my plate at once, makes it manageable.

# Teaching fundamentals vs the “shiny new stuff”

56:46

Alexey

**We have quite a few questions about the interviews. I just want to mention that Jeff will come to DataTalks.club again and will do a webinar about getting a data engineer job. That will actually cover that – mostly the interviews and how you need to prepare for that. So I apologize, but I will skip these questions, because there are a few questions that are related to actually teaching.**

**One interesting question is, “We learn the latest technologies, but most companies go for ‘tried and tested technologies’ (they prefer traditional versus new shiny tech). What is your opinion on that? Should we teach the new stuff? Should we teach the old stuff? Should we somehow find a balance? How do you do that?”**

57:36

Jeff

Well, you teach the fundamentals. I mean, the traditional tech is SQL and Python. Yeah, that sounds great. Even if you're a junior engineer, I'd say, if you can improve your Python skills – that's awesome. My first year as a software engineer, the other junior engineer next to me was just building Tetris and pure Ruby. And that's what the senior developer advised him to do “Just build arbitrary programming in Ruby. That's it.”

He got a job at Apple by the end of that year, so it seemed like good advice. I think you don't have to go super deep. Writing good Airflow code means that most of the code is in Python and is not relying on Airflow. Most of your skill set should be on Python and SQL. Like I said, that's what 85% of our course is, and then only probably like 15% is these shiny new technologies. But cloud computing – you can feel safe learning Docker and AWS. I think that's a safe bet. There are enough companies that are interested in that.

58:48

Alexey

**So if I want to teach data engineering, then I should teach SQL, Python, cloud computing, Docker, and that probably is the 20% that covers (Pareto principle) that covers 80% of the work.**

59:03

Jeff

Yeah. I mean, it depends who you're teaching it to. You know what I mean? You go to these Coursera courses and they're good. They teach you these skills, but they're not going deep into Python and SQL. Or they assume, I guess, that you already know that. But if you're taking someone who is not an engineer, or has not really worked with SQL that deeply before, you need to ramp them up on that intensely. That has to be the focus of the course.

# JigsawLabs.io

59:31

Alexey

**Okay. So it seems we're out of time. But maybe you want to mention anything before we wrap up?**

59:38

Jeff

Sure. So, we are accepting applications for a new course. The next course is June 15th. It's gone well. Everyone from our last cohort got a new job. The minimum salary was 100k, which for me, actually, isn’t particularly interesting. You know, I want their salary just to be good, but I just mainly care that they are launching a new career. Then this course also went well – they just graduated a couple of weeks ago, but we had a student get a job and has been a data engineer for the past couple months now. He got employed before he graduated. You can go to JigsawLabs.io if you're interested.

# Finding Jeff online

60:21

Alexey

**Yeah, I was going to ask how they can find it. Can you send me the link? If people want to find you and ask a question, what's the best way?**

60:30

Jeff

Oh, yeah. They can email me: jeff@ JigsawLabs.io. They can also ping me on LinkedIn. Just do “Jeff Katz” and type in Jigsaw. Yeah. Alexey, if you can just save the questions about the interview stuff, I can make sure that [cross-talk]

60:46

Alexey

**Yes, I will. I am going to send you these questions. They are about system design. They're about something else. I definitely saw a question about system design, testing pipelines, and some other stuff. Yeah. There are some questions related to teaching, which certainly we could not cover. But I'll send you them as well, so you have them.**

61:10

Jeff

Great. Thanks, man. That's great.

61:14

Alexey

**Thanks, everyone, for joining us today, for asking questions. Thanks, Jeff. By the way, when you look for Jeff in LinkedIn, there is another Jeff Katz, so be careful. There is one in Berlin – this is not right Jeff. I made this mistake once, so now I have two Jeffs in my LinkedIn network. [laughs]**

61:34

Jeff

Well, I’m glad he accepted you.

61:35

Alexey

**Yes, he did. So, he's my first level connection. He's also in the data space – so that was also very confusing for me. Anyways, yeah. Thanks for joining us today and have a great weekend!**

61:51

Jeff

Thanks, man. You too. Take care.