EPISODE 858

[INTRODUCTION]

[0:00:00.3] JM: Apache Airflow is a system for scheduling and monitoring workflows for data engineering. Airflow can be used to schedule ETL jobs, machine learning work and script execution. Airflow also gives a developer a high-level view into the graph of dependencies for their data pipelines.

Chaim Turkel is a Backend Data Architect at Tikal. He joins the show to discuss a case study of using Airflow to re-architect the data engineering workflow of a complex financial application. We discussed the problems that Airflow solves and the process of porting existing workflows to Airflow.

Before we get started, I want to mention a few updates from Software Engineering Daily land. I'll be attending a few conferences in the near future. Datadog Dash, July 16th and 17th in New York City, the Open Core Summit, September 19th and 20th in San Francisco. Also, we've got a new Software Daily app for iOS; it includes all 1,000 of our old episodes, as well as related links, greatest hits and topics.

You can comment on our episodes. You can have discussions with other members of the community and you can become a paid subscriber for ad-free episodes at softwareengineeringdaily.com/subscribe.

[SPONSOR MESSAGE]

[0:01:33.4] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust.

Whether you are a new company building your first product like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals. Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer.

We've also done several shows with the people who run G2i, Gabe Greenberg and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack and you can go to softwareengineeringdaily.com/g2i to learn more about G2i. Thank you to G2i for being a great supporter of Software Engineering Daily, both as listeners and also as people who have contributed code that have helped me out in my projects.

If you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW]

[0:03:25.0] JM: Chaim Turkel, you are a Backend Data Architect at Tikal. Welcome to Software Engineering Daily.

[0:03:30.5] CT: Thank you. Welcome to you too.

[0:03:32.7] JM: I just got back from a talk that you were giving about Airflow. I thought it would make for a great podcast. At the beginning of the talk, you were evaluating a post-mortem of a project that you worked on. This project started out with a application that had been wired together from a bunch of different applications, a bunch of different services, mostly on Google cloud. The gist of it was that after wiring together all of these different serverless services, the company that you were working with, the project that you were working with began to lose sight of the bigger picture of what they were working on. Give me the overview of this project that you were working on when you first encountered it.

[0:04:24.2] CT: The project, actually it's a very interesting evolution of what happened and I think it happens to a lot of startups, that they start using all sorts of technologies and without thinking for actually the long run. This specific project is a company called Behalf, where they do financing and help smaller companies finance themselves and their customers. They decided to use Google as their main platform, where the first issue there was BigQuery. They decided they want to use BigQuery as their major platform for all the data analytics.

From there, they started evolving their startup. They start adding all sorts of technologies as needed. For instance, they added Apps Scripts. Apps Scripts is another technology of Google, which is very easy to use. You write a few JavaScripts, the JavaScripts you add a few crons and they start working.

Now it actually brings me the end-product that I want, but there's no visibility in the process. I have to start thinking, "Wait a minute, this cron runs once every hour, so now I need to create another one every hour and a half, so that I know the first one finished before I write my next one. Then I have this product that evolves, because I keep on adding more and more tasks on crons, but I have no idea what's really happening." The biggest issue was when one of them fails. I have a failure, I fix it, but now what did I miss? What other processes in the whole flow did I miss? I don't know that I need to rerun them in order to fix, because of my previous data that was missing.

[0:05:58.2] JM: When you're talking about cron jobs, these are scripts that are written in Python, or node and they're just doing random operations at certain intervals, maybe every couple hours, or every three days, or every couple weeks. Is that what you would define as a cron job?

[0:06:14.6] CT: A cron job is a scheduling. I think the whole area of crons are a bit problematic, even Google itself has a lot of solutions for how to run crons. The specific ones, they call the triggers, they didn't call in crons, but it's identical to a cron. You give it the name of your Java function and you say, I want it to run every Sunday at 8:00, or every three hours, or every hour and a half. Then it triggers your process, or your specific function that you wrote.

[0:06:42.3] JM: In this application, what were the responsibilities of the cron job? How mission-critical were these cron jobs?

[0:06:51.5] CT: Part of the evolution, I believe, they thought at the beginning it wasn't mission-critical. Therefore, they were very lax with it. As the product evolved, they figured out wait a minute, this is very mission critical. We were using it to figure out again, Behalf is actually something similar to a bank. They give out money and it's very important that all my data is up-to-date and direct.

The project that I started on was actually considered not production, but offline analysis. Very quickly, we figured out that no, it's production-grade. If it falls down, we need to know immediately. Because we're using all the money to decide how much money to give back, or how much loans that we have to take out. We used to do all these batch processes, dependent on other data and one day, it fell. It stopped working. The this morning I came in, hysterical. It's not working. We're in big trouble. We're not production. We're just some batch line. Then it synced in, everybody understood. No, this is really production and we have to be production.

[0:08:00.9] JM: Sorry. I don't know how specific you can get into this, but these batch jobs were determining whether the company had to borrow money from a bank, for example? These were like, life and death, how much money is in our bank situations.

[0:08:18.5] CT: The way, a lot of financing companies, you take money and then you give it out to other people. I need to know which of my clients owe me money and I need to balance out my accounts every so often and the regulations that I have to stand by. I have to show that I'm up to standard of how much money I have in my bank account, as opposed to how much I'm allowed to give out. These are all sorts of standards that I have to do and I have to upload also my statements for somebody to validate and say, “Yes, we're working according to regulation.”

[0:08:52.3] JM: Now you could imagine a world in the distant future where every time there's a transaction, for example every time I issue a loan to a customer, or I give out money in some

FinTech application, it would update all of the different records, it would update the whatever regulatory compliance system is overseeing my incremental issuance process. We are living in a time where that is not yet possible. We have to do these incremental updates to one part of our application and then we might have to do this batch job to bring the holistic view into resolution across our entire application. I believe this used to be called the lambda architecture. Is that correct?

[0:09:36.2] CT: You brought up a lot of things. Yes. To begin with, batch is today the new bad word to use. We want to go to streaming. It's very interesting, lambda architecture in my opinion is in the past. We want to get rid of it. Google –

[0:09:51.9] JM: Keep dreaming.

[0:09:53.5] CT: No, there's already a product, Apache Beam. I don't know if you've heard of it.

[0:09:57.0] JM: I have. Yes.

[0:09:58.4] CT: Apache Beam on their post, they're out to kill the lambda architecture. That's what their mission is. Once we move to streaming, it is possible. It's a switch in our mind of how to use the data and how to process the data, but Apache Beam is pushing very hard to get rid of the lambda architecture.

[0:10:16.1] JM: I mean, Kafka's been trying to kill the lambda architecture for a long, right? Oh, Jay Kreps with the Kafka architecture. We're killing the lambda architecture. Isn't the lambda architecture just a byproduct of relativity? It's just you've got one area of the world that updates and it takes a while for the rest of the world to notice that that update has occurred.

[0:10:38.1] CT: The problem is technology of how to know about my late data. The biggest problem and that's why they invented lambda architecture is if I have data that came in late, I now need to retroactively fix it. Apache Beam has given us an API to do it. I've even read if I remember correctly, that even Kafka is thinking of adopting the Beam interface to help them solve part of those issues.

[0:11:04.3] JM: I know, I was such a Beam skeptic at first. I didn't understand what they were trying to do. Maybe, I guess I still don't understand what they're trying to do exactly. I know you have Apache Beam, which is – actually, okay, I'm just going to ask you. Describe to me what Apache Beam – and by the way, we're getting very far from Airflow. We will return to Airflow eventually, but let's go through the data engineering buzzwords one at a time. Explain to me what Apache Beam is and why it's serious, or why it's significant.

[0:11:33.4] CT: Apache Beam started in Google as data flow of Google. Then they put it on the market as an open source. Whoever hasn't read, I'm very advanced that you should read their two articles; 101 streaming and 102 streaming. The idea behind Apache Beam was I want a vision how we should do streaming. If we look at Spark, or Flink, or any of the other platforms that are out there, they give you a platform and the platform slowly evolved according to the needs of the market.

Apache Beam said, "Let's stop. Let me think what I want. What are the issues? The issues are I have an event that happens." There's always two-time bases on every event; there's process time, there's event time. Nobody put that on the table until Apache Beam. Everybody knew it and they knew that's an issue we need to somehow deal with, Apache Beam put it on the table. They said they're two times in the API and the SDK that I give you, you can reference either one. You can decide which one you want a reference.

Of course, the byproduct of the difference between the two is how do I deal with late data? When do I trigger my windows? Do I trigger my windows on a database, on a time base, or on late database? They started inventing abstractions. They said, "Let me give you a vision of how we want the streaming to work."

Even if Apache Beam does not succeed, they're pushing the whole market. They're forcing the market to rethink and reput in all these actual attributes in their system and they're changing everybody. Even if they specifically don't succeed, I think they're going to change everything.

[0:13:14.3] JM: When Apache beam came out, it was – the way I understood it was a way for other streaming frameworks to write ways for their data pipelines to be translated. I'm sorry, it was a way for any Beam-compliant definition to potentially be compiled into Spark, or Flink, or

Storm, or data flow, accepted data flows proprietary, or still is proprietary, so this is what always confused me about it is Beam is this open way for you to compile any data pipeline into one of these respective streaming back-ends, but you ultimately are in many cases going to want to run it on Google dataflow, even though its proprietary. Although I guess, that's because – it's not because Google wants to lock you into their proprietary system, it's probably because data flow itself is too tightly coupled to Google infrastructure, so they haven't been able to open source it yet. This is eventually going to become an open source thing, right?

[0:14:19.6] CT: The way I understand it, Beam has done a few things. In addition to what I described of the vision of how streaming should work, they've also decided to do another level of abstraction. In order for them to be relevant, they could have just done data flow and then they wouldn't have been relevant, because people will say, “I don't want data flow. I'm Spark.” They decided to do another layer of abstraction and say, “We'll give you an SDK, some DSL language. You write in in the language that you think of streaming and we will then write runners, which will run it on a proper platform.” They've now taken it to the next level and they've said, “Wait a minute, I want another abstraction. The current SDK was in Java.” They say, “Wait, why Java?” Python was also added early on, but now they've said, “No, we want to take it totally different. You're a Go shop, write it in Go. We will then translate your Go to our layer intermediate and run it on Spark, or maybe you're going to write it in Java SDK and run it on Go.”

[0:15:24.3] JM: The idea in any of these cases is that you're writing it in Beam.

[0:15:29.9] CT: You're writing it in the Beam concepts and then you will choose the runner according to your performance issues, or according to your do you need an on-premise on the cloud solution. The idea is you write your abstract thinking in the language that you're used to and in the Beam concept. Then the second stage is okay, now where do I need to run it?

[0:15:51.1] JM: Sorry, so there are frameworks for doing this in Go and Python and Java and everything and you have the Beam idioms in all of these different frameworks?

[0:15:59.8] CT: They're adding more and more frameworks that I can write the Beam concepts.

[SPONSOR MESSAGE]

[0:16:13.4] JM: As a software engineer, chances are you've crossed paths with MongoDB at some point, whether you're building an app for millions of users, or just figuring out a side business.

As the most popular non-relational database, MongoDB is intuitive and incredibly easy for development teams to use. Now with MongoDB Atlas, you can take advantage of MongoDB’s flexible document data model as a fully automated cloud service. MongoDB Atlas handles all of the costly database operations and administration tasks that you'd rather not spend time on, like security and high availability and data recovery and monitoring and elastic scaling.

Try MongoDB Atlas today for free, by going to mongodb.com/se to learn more. Go to mongodb.com/se and you can learn more about MongoDB Atlas, as well as support Software Engineering Daily by checking out the new MongoDB Atlas serverless solution for MongoDB. That's mongodb.com/se. Thank you to MongoDB for being a sponsor.

[INTERVIEW CONTINUED]

[0:17:36.1] JM: Bringing us back to the application that you were working on at this FinTech company, so give me a little bit more of a description in terms of how this FinTech company had wound up in a situation where it had this sprawl of tools and this problem with these essentially a lambda architecture?

[0:17:55.7] CT: I think again, I don't want to sound that I'm so much against serverless, because I'm not. It's a matter of how you use the tool. The way the tools, the way that they're selling it so aggressively they want you to get into it, they make it too easy. The problem is you're like, “Okay, I need some function to do it.” Within a few minutes, Google will say, “No problem. Here, just write your function here and it works,” and it does work. Then you start hitting the limitations. In App Script specifically, a specific script cannot run for more than six minutes. If you have a long process, you're stuck. The overall time of all your scripts depending on your projects can't pass a certain amount of hours.

You start hitting all these limitations, but you started out because it was so easy. We needed monitoring, so the CTO said, “No problem. Google Spreadsheets. You can write JavaScript behind it. It connects to BigQuery, reads the tables, within a few minutes, he had a spreadsheet up and running with information.”

[0:18:57.2] JM: He was using spreadsheets for monitoring?

[0:18:58.6] CT: Google spreadsheets for monitoring.

[0:19:01.5] JM: That’s strange.

[0:19:01.6] CT: Why? Because it's so easy to do. That was the problem. It was too easy. Then at one point you're like, “I've lost everything. I don't know where anything is, because I did it very quickly. I didn't put any much thought into it of who's going to use this, how are we going to use it, how are we going to scale.” These are issues that startups at the beginning don't think, which is okay. It's legitimate. At a certain stage and scale, you have to rethink and then find the proper solution.

[0:19:31.5] JM: How old was this company?

[0:19:32.8] CT: This company is three-years-old.

[0:19:34.6] JM: Three-years-old. Okay, wow. Are they entirely “serverless,” or do they have servers running for any significant application?

[0:19:42.9] CT: They have. They have a lot of servers. The company I would say is divided between the R&D, which is all in Amazon and they have microservices and Java and Spring and the whole stack, and then there's the BI area. The BI area and the machine learning, they all sitting on Google with the BigQuery.

[0:20:00.1] JM: I have seen that architecture in many different places where people are like, “All right, well we trust Amazon as the place to run our business logic, but the Google API is for data processing are so amazing that we want to have this dual cloud set up.”

[0:20:26.5] CT: Dual cloud is very difficult, but at least on the BigQuery I haven't actually worked with redshift. I've read about it. From what I understand, it doesn't really compare to BigQuery. The ease of BigQuery and the performance of BigQuery is just unthinkable. You just throw as much data as you want, you run your queries, it'll give you an answer. It's not real-time. It's not PostgreSQL, but you'll get an answer and you don't have to think about it. You don't have to think about how many machines I have. I don't have to manage my machines. I just put the data in, that's it.

[0:21:01.0] JM: It connects to Google sheets.

[0:21:03.5] CT: It connects to Google sheets.

[0:21:06.3] JM: Can you help me understand that a little bit more? You said they are doing monitoring with Google sheets. What does that mean?

[0:21:14.1] CT: That means you have a sheet, one column is the name of the table that I need to synchronize from some external source, I then do a select on that table for a max time, I see when was the last time it was synchronized. With formulas in Google sheets, I compare it to the current time. If there's a discrepancy of an hour, or half hour, depends what we decide, there's another column that says to which e-mail send it. Then in JavaScript behind the Google sheet, you iterate over the rows in the spreadsheet and any discrepancy, you send an e-mail with Google Apps.

[0:21:51.3] JM: An example of – this would be like, financial discrepancies.

[0:21:54.0] CT: Not necessarily. The discrepancies here is data synchronization. We have a lot of external sources, whether it's Salesforce, MongoDB, PostgreSQL, and we have the data flow, Google Beam that we use to import and ingest all these data. Now if the Google's flow is down, not because of Google, but because we are not running it, currently we're in batch mode, not streaming. We have to run every half hour. If whatever reason something got stuck, my data is not being updated and I need to know that as fast as possible.

[0:22:27.3] JM: Interesting. They were basically using the spreadsheet as I see, as monitoring, as a way to identify data discrepancies and then kick off an e-mail to an admin. Basically, the admin would have to go in and solve it. Okay, so I'll tee you up. What is wrong with this process?

[0:22:51.8] CT: It's not managed. I think that's the main word, it's just simply not managed. There is no software engineering practices in the way it was built. It took me a while to actually put my finger on what bothered me the most, but I think that was the biggest issue. On the production side, it's the management. We don't know what's really happening. Even if my spreadsheet tells me, "Look, this failed." I'm like, "Okay, it failed. Now what?" "It's a problem at the ingest." "Okay, I fixed my ingest. What does it apply to? In which other tasks now do I have to fix, because I haven't been ingesting for two hours?"

I have to try to figure out the whole flow, which is cron-based in I don't even know which systems all the time, so we're not managing it and our codebase is not – there's no software architecture, classical paradigmas. I don't have git. None of my code is in source control. It's inside these spreadsheets of Google, inside Apps Scripts. I want to manage it. I need versioning. How do I know if I make a fix? My spreadsheet has a bug. You go in and you fix it, you think you fixed it, but you broke something else. There's no unit testing, there's none of the standard paradigmas we know as software engineers.

[0:24:06.6] JM: Let's say I am one of these admins. I get an e-mail from the spreadsheet system, what is my life like for the next hour and a half, or two hours? What am I doing?

[0:24:19.0] CT: You're sending out a lot of Slacks and WhatsApp, "Please help. Please help."

[0:24:22.8] JM: Please help, to who?

[0:24:24.7] CT: To me.

[0:24:26.6] JM: Okay. Why doesn't spreadsheet do that? Okay, so to explain, so then what do you have to do? How are you debugging the system when you get one of these e-mails?

[0:24:36.2] CT: I'm trying to figure out which system. Again, we started at one point when we figured out it was too much for me to handle, so we started writing these documents. If this happens, go to here. If that happens, go to there. It might have been the Google App Engine, because I was deploying data flow with Google App Engine. Maybe my Google App Engine stopped working, or maybe the deployment that I did in Google data flow, something got stuck there and therefore, when I run my batch jobs, I make sure that I don't have two of them running at the same time.

If one job got stuck for whatever reason, I'm not going to run any other ones, so the whole pipeline gets stuck. We start looking through all the different logs that we have, because all our logs we took into Sumo Logic and we do log analysis there. Then I have to try to figure out where is my problem.

[0:25:28.5] JM: Okay. At a certain point, you must have taken a step back and said, "This is madness. What am I doing? How has my life come to this?" You said to yourself, there must be some alternative workflow.

[0:25:45.6] CT: Yes. I think the hardest part was when I fixed my problem, the system was down for 12 hours. We had a lot of issues. We fixed it. I came to the CTO very happy. I told them, fixed, everything's running. You can feel okay.

[0:26:01.3] JM: This was after a spreadsheet e-mail?

[0:26:03.1] CT: Yes. We had a lot of spreadsheet e-mails. He said the whole system's not working. I worked on it. It took me a while to fix. I was happy and I saw in his face something bothered him. What's the issue? He said, "I understand yours works, but now I have to figure out what else do I have to fix because you haven't been running for 12 hours." We sat for three days, three days to figure out what processes are affected for my downtime.

That was the point that I said, too much. If the CTO doesn't know what was affected, that means nobody in the company knows. He's a hands-on guy. He knows everything that's happening. I asked him, "Okay, draw me the flows." We didn't know how to draw the flows in the system, because nobody had it in front of them. It was a patch on patch. I know how to run after this. He knows how to run after that. A goes after B after C, but nobody knows that when C falls, now we've got to backtrack all of them.

[0:27:05.8] JM: Okay. I think this leads into Airflow nicely. Let's take a step back. Tell me what your knowledge of Apache Airflow was up to this point. Had you worked with Airflow at all, or were you conscious of Airflow?

[0:27:23.5] CT: I knew of the name Airflow. I had never used it and I have never written in Python until Airflow.

[0:27:30.0] JM: Okay, well actually let's just cut to the chase. Why was Airflow a useful solution to this problem of not being able to visualize the flow of data, the flow of actions that were being taken across the application?

[0:27:45.6] CT: We understood that we need an orchestration. We need something that would switch out the whole cron business. Again, I also wanted everything out of the serverless, as far as I want everything in my git process and I want to manage the processes. I need some orchestration that will manage my process. I want to know that A is running after B, because I want to visualize it and I want them dependent on each other. If A stops, I want B to stop. Then when I fix A, B will automatically run and I won't have to think afterwards, wait, what's happening here? A has not been running for a long time, but B continues to run, it has invalid data.

We were looking for an orchestration and they're not a lot out there. There are. Airflow has the highest hype. The differences between the two are so which language are they're written in. There's Luigi that's written in Java. It hasn't really been updated that much. Some of the configuration files are still XML. We were looking for something that was more advanced. I was in a BI team, which the BI team do not really know how to write Java-based and all sorts of standard languages. The script language was much more natural for them.

The Python went together and the Python also gives me an advantage of high deployment. In Java, you have to compile it, you have to package it and then deploy it. Python with Airflow, you just copy a file to a folder and it takes –

[0:29:17.9] JM: You would describe Airflow as a workflow orchestration tool?

[0:29:21.5] CT: That is how Airflow defines themselves. Yes.

[0:29:23.7] JM: Okay. What is a workflow orchestration tool?

[0:29:26.8] CT: An orchestration tool means, and people do confuse it a lot with Jenkins, for instance. The orchestrator does not do the actual lifting of the data, or you don't want it to do it. You want it to orchestrate. If I have a data flow process and after my data flow process, I want to run different queries in BigQuery, Airflow is classic. It knows how to organize it, first to step A. If step A works, then do B, then do C. The actual processing of ABC is not done. The data itself, I don't bring it into Airflow.

I trigger other processes. I can bring the data to Airflow, but it's less preferable. That's the definition of the orchestrator is it calls different things and it combines. If for instance, I need data from Amazon to Google, I can very easily bring it. Here, I do need to lift the data because it's between clouds, so I'll have to bring the data into the Airflow from Amazon and then push it to Google.

Part of the advantages of Airflow is their operators, which is open source, so all the providers have already written their providers in Airflow. With a few bits of code, You can take from Amazon, from S3 and push it to Google and you don't have to write that code. It's already there. You just have to combine it and build your puzzle and then you have your whole orchestration.

[0:30:53.9] JM: When did you realize that Airflow was going to be a useful tool? Or did you do a proof of concept with Airflow to try to figure out if this was going to solve your problems?

[0:31:05.4] CT: I asked around. Also, we're part of Tikal, we have these meetings once a week and once a month. I raised my issue and people brainstorm and throw all sorts of ideas, have a look at this, have a look at that. I did some research and I did understand that we're looking for some orchestration. We also needed – one of the requirements was that it should be a server. We didn't want to manage it ourselves, so I was looking for a SaaS solution.

Airflow is also very hype. We also wanted to make sure that it's open source and that I'm taking a winning horse. I don't want to take some product that after half a year we have no support and it's dead. All of those combined, Airflow seem to be the magic solution. Again, no silver bullet, nothing solves all your problems, but it did solve I would say 70% to 80% of our issues.

[0:32:01.8] JM: A little bit of background on Airflow, we've done a couple shows with Max who started Airflow. We did one back in the day when he was at Airbnb. That's where the name comes from, Airflow, Airbnb. Yeah, Airbnb obviously has tons of data engineering problems and Max had a background of having worked at Facebook, solving data engineering problems there, so then at Airbnb he encountered similar problems to what he encountered at Facebook and was able to get the Airflow project off the ground. Then when he went to Lyft, obviously Lyft has probably even more data engineering problems than Airbnb. He continued to work on and support the Airflow community to some degree.

What I like about Max is speaking of full stack, he really has a sense of from the lower layers of data infrastructure, to the higher levels of how an operator wants to be using a data infrastructure support tool, I guess you would call it. The UI, the UI for Airflow – I mean, I've never used Airflow. I've just seen screenshots of it and videos of it and stuff. It looks like the thing is designed well from top to bottom. It seems that's part of why an ecosystem is really developed around it.

[0:33:19.6] CT: Yes. I think it also has to do with as you said, it was developed by people that had felt the issues at hand. I think almost every company has invented a small version of Airflow. Everybody has written some cron to run things. Most of them stop at the engine level, to actually bring it to distributed and graphs where I can also understand what's happening. I'm not sure anybody except for Airflow's actually gotten to that level. They still have a lot of work. It's not the end product.

[0:33:55.4] JM: Right. In most of these cases with the tool, it's like the world figures out that there is a need for a tool, like a workflow orchestration tool. Then at some point, people start talking and then everybody realizes, "Oh, dang it. Google built something internally five, 10 years ago, right?" In your archaeological dig of the world of workflow orchestration, did Google have something? Did Google make something that was like, do they have a workflow orchestration tool, or have they solved this problem in some other way?

[0:34:28.2] CT: I'm not aware of a tool from Google. I do remember from my past Microsoft, about 10 years ago tried to do something like that. They even visualized it. It's actually very similar to a lot of other tools that are flow diagrams. You have a flow. You draw your flow and then you insert serverless codes functions for each flow, but I think also the Python made a very big switch. We're seeing a lot of tools with the same architecture as Airflow, where they have a platform that's built on Python and the community can add functions and it's very easy to be a pluggable architecture.

I'm not sure if any of the others have something like that and they've built other solutions that some of them I'm aware of. Sometimes it takes time. It was like Facebook. At the beginning, everybody was like, "What do I need that for? It's useless." Today, nobody can move without it. It took a few years till people understood the effect that it's having on the whole community.

[0:35:35.4] JM: Okay, let's make this concrete for people. How did Airflow solve some of the problems that you were encountering?

[0:35:45.6] CT: Airflow forces you to think of the flows. The building blocks of Airflow are a DAG, that's the Direct Acyclic Graph. I'm thinking graphs. As I said, it's very easy. They even, advance they say, put your configuration in the Python itself. Don't separate configuration from Python. It's the same thing, because you're an orchestrator. You stop thinking about all the tools and you're thinking about your processes. What's the first thing I want to do? I need to bring my data. Okay, I'll trigger the data flow. When that finishes, what's the next process that I want to do?

When I sat with the CTO, he was like, "Okay, put an Airflow." I said, that's the easy part. Now sit with me and tell me the order of the things that you want. It forced him to sit on a business level. You're right, what are the dependencies, the business dependencies that I want to do and let's set it up in Airflow. Then I showed him the dashboard and you just see the business right in front of your eyes, because there's a beautiful graph that shows it to you. You can now manage it, including logs in one place, including Slack notifications directly to me. I don't need somebody else to start coming into the loop. It's simplified a lot of the thought process more than the actual technology.

[SPONSOR MESSAGE]

[0:37:17.7] JM: DigitalOcean is a simple, developer-friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy, with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow.

DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support. Plus they've got 2,000 plus tutorials to help you stay up to date with the latest open source software and languages and frameworks.

You can get started on DigitalOcean for free at do.co/sedaily. One thing that makes DigitalOcean special is they're really interested in long-term developer productivity. I remember one particular example of this when I found a tutorial on DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence and an attention to just getting developers off the ground faster. They've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That’s do.co/sedaily. You will get started for free, with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:39:19.4] JM: Now you must have had some system that was managing the workflows from your transactional databases in Amazon and there was talking to the, I don't know, probably some ETL stuff in Google and some other data streaming stuff in Google. You had some work flow that was being orchestra. All right, electric guitars have started next door. Welcome to the live show of Software Engineering Daily. You already had some workflow infrastructure set up, some workflow data pipeline setup. Was the Airflow stuff subsuming that? Or was it replacing it?

[0:40:09.0] CT: Every company invents their small Airflow. We actually invented it twice; once in Java and once in Python. That was part of the understanding, this has got to change. The whole idea is to replace everything. We want one tool as much as possible. Everything was converted into Airflow. We stopped using all these other services and we're in the migration. We're at about I'd say 60% of migrating all our old processes onto Airflow.

[0:40:41.0] JM: Okay, so during the migration, just to be clear, are you replacing code that you had from the previous set of scripts and ETL jobs and stuff, or are you just importing that code into the Airflow world?

[0:40:59.0] CT: It depends on the ETL itself. All the ETLs that are SQL-based were just moving over. Some of the ETLs have actual logic that was written and now we're transferring it either to Python, or into the framework of Airflow. If the original was in Python, it's a bit of copy-paste. If the original was in Java, then some of it is being rewritten. The fact, we're moving to Airflow is also causing us to rethink. Sometimes the actual move to Airflow makes you redesign your systems and some of it which is delete, because we figured out we don't need it now.

[0:41:33.5] JM: Okay, so if I understand correctly, the world before you start adopting Airflow is you've got these cron jobs, the cron jobs get kicked off periodically. One of the cron jobs might be a Python script that says, okay it's been four hours. It's time to grab all the transactional data from Amazon and load it in BigQuery, do something in BigQuery and then you spend down that that Python service that was spun up to do that ETL job. In the Airflow world, you move that code into Airflow so that it's consolidated with all the other random cron jobs and stuff.

[0:42:08.2] CT: Correct. I would say the jobs are more or less divided into two. The majority are SQL-based, so it's ETL. I'm generating SQLs, which either test data, or transfer data. Then I have the whole area of reports. I'll actually go over the transactions and send out e-mails to our customers with some summary of what's happening with their transactions during that day and that we also do on top of Airflow.

[0:42:35.6] JM: Okay, cool. There is this process by which you need to move all these old scripts and cron jobs and stuff into Airflow. Is that a gradual process, or do you have to do that all at once and then and shift your entire workflow over to Airflow?

[0:42:55.5] CT: We debated about it. We obviously want it gradual, because it's too big of a risk to do it in one. The way we actually did it is by running them in parallel. The difference just was the target of the ETL. At the beginning, the new system we ran it on the side and all its results were written to a different target and then we compare the two targets; the current production and the new one, the staging on the side that's running the new code.

After we were very fairly comfortable that the results are the same, we flipped them. My old system still runs, but runs now to the temporary target. The new one writes to the production target. Then as we gain confidence, we're going to shut down the old system and put it to rest.

[0:43:44.6] JM: What problems has this solved for you, moving to Airflow?

[0:43:49.8] CT: I would say the main problem is visual. I know what my tasks are. I know the order of the tasks. Airflow has also changed our thoughts. In other words, Airflow introduces another concept, which is immutable. You want to run your task at a specific time and you want to be able to run it if necessary, five times. Because when you run a task, the import to the task is the timeframe.

Airflow manages that timeframe. Let's say for whatever reason, Airflow was down for two hours and I have a cron that runs every 5 minutes, it's not going to come up and run once. It's going to come up and run as if it ran every 5 minutes. For each one, it will give me what's called backfill; the timeframe of when it was supposed to run. Then I write my code based on that time and then I can backfill all my data according to the time that I was missing, which is a very big concept that we changed in all our ETLs. That solves the problem of back data, of incorrect back data. If I have a task that failed yesterday, after I fix it I can rerun it on yesterday's data.

[0:45:08.5] JM: Why is Airflow so helpful with these backfill issues?

[0:45:13.3] CT: Because by design and again, that's the whole issue of Airflow is the people that designed it, they designed it with the problems in production that they had. If I have a task that fell yesterday and it takes me a day to fix, when I fix it and rerun that task, it has to run in the context of yesterday. Every task that's running Airflow, the input to the task is the time frame. You never ask the computer what's the time. You don't say, “I'm running now.” No such thing. You get the time as an input parameter. Airflow manages your time.

[0:45:50.4] JM: What? Can you explain that in more detail? I’m sorry, I’m having trouble understanding this timeframe thing.

[0:45:54.6] CT: Most people when they write code, let's say have to write a select and I want to say, “Give me all the transactions that were done in the past hour.” I would say, “Let me look at the timestamp of the machines, subtract an hour and that I'll put in my where statement on the SQL.” I'm always looking at the last hour from the current timestamp. Airflow tells you, don't do that. I will give you the time. The time is the 14th of May, 10:00.

In my where statement, I'll take the 14th of May at 10:00, subtract an hour. I'm never saying, what is the time? I'm never saying, I'm running now. It says, run it 14th at 10:00. Then it'll say, run it 14th at 11:00. If 12:00 fail, even if I run it on the 16th of May, it'll tell me run this code as if you're on the 14th of May at 12:00.

[0:46:46.0] JM: When a failure occurs in an Airflow job, or I mean – how does Airflow deal with kinds of failures in the underlying system? If Airflow is orchestrating these different jobs, Airflow is not – Airflow doesn't know what's going on in the underlying scripts that it's kicking off. You could have errors and exceptions and things that might screw up your data pipeline. That can be really problematic if you have, depends on relationships across the DAG. How does Airflow handle failures?

[0:47:22.2] CT: They've actually given a framework as we keep on repeating myself a bit, but they've thought of a lot of these issues. That's why I like Airflow. For instance, you have different scenarios; one is that I have a failure, not because it failed, but as far as my business logic, I don't want to continue the DAG. I've done two tasks. I don't want the third one to run for a business reason. The way they've designed it is every task has a return value. Based on that return value, they evaluate whether or not to run the next task. I can return false in my task and it will not run the next task.

On the one hand, they're giving me a framework on the business level to handle not continuing my DAG. On the other hand, exceptions happen, errors happen that I did not think of ahead of time. They have built in any exception that happens, they know how to catch, and you can define by design, I want you always to try three times. Maybe it was just a connection error and if you'd retry it, it'll fix itself. If that still doesn't work, then they have other options for starters, the whole DAG will fail. You'll see it in the GUI. You have the option of callbacks. You can write a callback on the tasks level, or on the DAG level and write whatever you want.

For instance, we write every failure to a table in BigQuery for historical reasons and we send a Slack message using Airflow, of course. I do that generically on all my DAGs, because I have the same callback for all of them. The framework is very easy to enhance and add my own logic, which is a layer on the platform for everything that's written without even me knowing what's going to be written in every specific DAG, because we've decided all DAGs that fail, they need to be written to a database and send in Slack.

[0:49:19.0] JM: We're running up against time here. What other reflections do you have about the state of the broader data streaming and workflow orchestration and data pipeline space? It's clear that this project was a great case study in the state of the art of average data pipelines, but any other broader reflections just in tandem with your project, from other stuff you've been reading, or stuff you've been studying?

[0:49:52.3] CT: From what I see, it's the integration of data sources. Obviously, we talked about Apache Beam, so the streaming is a whole industry that's moving very forward. There are a lot of technologies out there for the streaming, whether it's Apache Beam, Kafka streams, Spark streaming. Then we have Airflow that's gapping another area, which is more of an area of orchestration of different services. Somehow, we need to integrate all of these together. That's still a missing part.

I see it also in the cloud, they're trying to gap how do I connect Amazon and Google together as far as the services on a service layer? Google is coming out with some service that I can connect and interconnect them. As companies grow, their customers might demand, I want our services to run on Amazon and a different one on Google and maybe some of them even on-premise. To be able to orchestrate different clouds and different technologies is becoming more and more complicated.

[0:50:55.4] JM: You were working on an application that was three-years-old, which is a very new application. It sounds they had some – even though you were getting paged by a spreadsheet that was doing these data discrepancy monitoring, that's still a futuristic cool problem to have. I mean, most people do not have the integration set up to do something that sophisticated, to set up this spreadsheet workflow. That's pretty different and unique for many companies, like you think about legacy insurance companies, legacy banking companies, legacy agricultural technology companies. They've got these really old systems and there's a perennial problem that they're dealing with over the last, I guess five or 10 years and particularly more acutely now that they all want to move to “machine learning,” the problem of the data platform.

They want this, at least as I understand it, at this data platform, this place where I can go and get data from any place in the company. I can set up a stream from it easily. I can do machine learning on that data really easily, because this is where so much untapped value lies and like an insurance company has decades and decades and decades of old data. It would be so great if their data scientists could access it and do interesting things with it, while also being in compliance.

There seems to be a vision for this unified data platform. One thing I'm trying to figure out is will there be this unified data platform that sits over everything, like your data operating system, or is it just going to be this patchwork of tools that we're going to gradually cobble together over do you got Airflow over here, you got Beam over here, just patching these things together and slowly figuring it out.

[0:52:40.6] CT: If I understand you correctly, I think the word is data lake. That's the new buzzword that they're trying to get to. Also Google, I've seen and also Amazon, they have antenna and all sorts of products, more listed to catalog. It's a catalog of my data.

[0:52:57.2] JM: Data catalog.

[0:52:57.9] CT: I have all my data in this data lake, which is more or less storage. I put it all into the storage unstructured and then I'll have pipelines that will structure my data and give me also it's a materialized view. They're starting to come out with platforms, but the problem here from what I see is the added value is only when they know what the data is. All these platform, all these catalogs, they try to scrape metadata on your data from the name of the file from the structure and then they try to help you out.

There's actually what we're missing is a Google index for my data lake. Today, I can search in Google and find whatever I want very quickly on the internet. If I have my data lake, there's no easy way to just search and find things there. Data discovery I think is now one of the biggest buzz words that they're trying to solve.

[0:53:51.1] JM: Have you seen any data discovery platforms that have been interesting to you?

[0:53:55.2] CT: I know Google, again, they're all starting. They're at the very beginning. I think something, the actual word discovery just reminded me. I think Google has something called discovery, but they are very beta. They're at the very beginning, where they give you basic information.

They have all these catalogs where they know how to scrape the information and run the information and try to take and extract. You can map them and then you can run SQLs if you know the structure, but I still need something much more intuitive. That's I think we feel we lost something when we went to the web. When I had all my information on my desktop, we had then Microsoft – I don't remember the name. They had some indexing that indexed your computer.

I had all sorts of programming that we're using Lucene index to index your computer, so you don't have to remember where anything is on your computer. You just put in a text and it finds it, it's a PDF or Word document, whatever. I need that for my data lake.

[0:54:52.1] JM: Something you should look into, we did a show recently about Amundsen, which is a data discovery platform out of Lyft. It's basically what you're describing, it's an index. Lyft just has so much data, right? Because it's a ride-sharing platform. I'm sure your Ube has even more data, because they're across a larger swath of area. It is what you're describing. If I am a data engineer, or an operations person, or a data scientist and I'm looking for cool datasets to utilize within my company, I enter a little search query in Amundsen and I can find the datasets I need and figure out the permissions for them.

Anyway, I mean, I'm with you. It does seem to be an emergent piece of the data platform. Well Chaim, it's been really great talking and I'm very grateful for the talk that you gave and a time we've had to talk together.

[0:55:43.7] CT: Yes. I've actually enjoyed it very much. It's my first podcast.

[0:55:46.9] JM: Okay. You're a natural.

[0:55:49.8] CT: Greatly enjoyed it.

[0:55:50.9] JM: Okay. Thank you very much.

[END OF INTERVIEW]

[0:55:56.3] JM: Commercial open source software businesses build their business model around an open source software project. Software businesses built around open source software operate differently than those built around proprietary software.

The Open Core Summit is a conference for commercial open source software. If you are building a business around open source software, check out the Open Core Summit, September 19th and 20th at The Palace of Fine Arts in San Francisco. Go to opencoresummit.com to register.

At Open Core Summit, we'll discuss the engineering, business strategy and investment landscape of commercial open source software businesses. Speakers will include people from

HashiCorp, GitLab, Confluent, MongoDB and Docker. I will be emceeing the event and I'm hoping to do some onstage podcast-styled dialogues.

I am excited about the Open Core Summit, because open source software is the future. Most businesses don't gain that much by having their software be proprietary. As it becomes easier to build secure software, there will be even fewer reasons not to open source your code.

I love commercial open source businesses, because there are so many interesting technical problems. You've got governance issues. You got a strange business model. I'm looking forward to exploring these curiosities at the Open Core Summit and I hope to see you there. If you want to attend, check out opencoresummit.com. The conference is September 19th and 20th in San Francisco.

Open source is changing the world of software and it's changing the world that we live in. Check out the Open Core Summit by going to opencoresummit.com.

[END]