EPISODE 1290

[INTRODUCTION]

[00:00:01] JM: ELT is a process for copying data from a source system into a target system. It stands for extract load transform, and it starts with extracting a copy of data from the source location. That data is loaded into the target system like a data warehouse, and then it's ready to be transformed into a usable format for things like modern cloud applications. The company Meltano provides code that manages ELT pipelines through an open source, self-hosted, CLI-first, debuggable and extensible process. Meltano projects manage your Singer tap and target configurations to easily select which entities and attributes to extract. These pipelines track their own incremental replication state so they can pick up where the previous ones left off. Once your raw data is in its target source, Meltano helps you transform it into a usable format. These pipelines can run on a schedule and they can be fed to supported at orchestrators like Apache Airflow. In this episode, we talked to Douwe Maan, a cofounder of Meltano, about their product market fit and delivery plans. Meltano is an ambitious project. And I hope you enjoyed this episode.

Our first book is coming soon. *Move Fast* is a book about how Facebook builds software. It comes out July 6, and it's something we're pretty proud of. We've spent about two and a half years on this book. And it's been a great exploration of how one of the most successful companies in the world builds software. In the process of writing *Move Fast*, I was reinforced with regard to the idea that I want to build a software company. And I have a new idea that I'm starting to build. The difference between this company and the previous software companies that I've started is I need to let go of some of the responsibilities of Software Engineering Daily. We're going to be starting to transition to having more voices on Software Engineering Daily. And in the long run, I think this will be much better for the business, because we'll have a deeper, more diverse voice about what the world of software entails.

If you are interested in becoming a host, please email me, jeff@softwareengineeringdaily.com. This is a paid opportunity. And it's also a great opportunity for learning, and access, and growing your personal brand. Speaking of personal brand, we are starting a YouTube channel as well. We'll start to air choice interviews that we've done in-person at a studio. And these are high-

quality videos that we're going to be uploading to YouTube. And you can subscribe to those videos at YouTube and find the Software Daily YouTube channel.

Thank you for listening. Thank you for reading. I hope you check out Move Fast. And very soon, thanks for watching Software Daily.

[INTERVIEW]

[00:03:03] JM: Douwe, welcome to the show.

[00:03:05] DM: Thank you, Jeffrey. Very happy to be here.

[00:03:07] JM: Meltano. Last time I talked to somebody about Meltano was a few years ago. It was a project in GitLab basically like a project that was in a laboratory deep in the bowels of GitLab where very few people were using it. But the hypothesis was powerful. And the hypothesis was basically, if you look at what GitLab did for DevOps, we have an opportunity to do that for data engineering. We have an opportunity to unify disconnected tools into a cohesive workflow that is replaceable and modular, yet slightly opinionated. What have you learned in the last few years that caused you to have enough confidence in the project to spin it out as a separate company?

[00:03:50] DM: Yeah, that's a great question. So you interviewed Danielle Morill, who was the general manager for Meltano at the time, about two years ago. And from the beginning, our conviction around building Meltano has always been that, just like in GitLab, pulling together all these different steps of the DevOps lifecycle into one platform that can kind of become better than some of its parts with really great integration between all these different tools that might be used by different people on the team, but getting them together in one environment. We thought that similarly, in the data space, while there exists many point solutions that are really great both on the kind of closed source proprietary side and in the open source ecosystem, there was a significant opportunity for better integration between those tools, enabling better collaboration between team members across different disciplines.

So from the beginning, the idea has been that GitLab's data team wanted this kind of data tooling that embraced those DevOps best practices and principles that are so key to Gitlab's way of working. And we thought that if we pick some of these best in class open source tools for data integration, transformation, orchestration and visualization, bring them together in one platform that provides the glue, we would build really compelling end-to-end data lifecycle solution for teams, not just GitLabs.

So a few years ago, when you last spoke to Danielle, we were essentially working to realize a proof of concept of a single platform that pulls together these technologies, and has this kind of plug in architecture where you can bring in DVT for transformation, Singer, depths and targets for integration, Airflow for orchestration. And this would make it kind of all C function as a cohesive hole, one repository for a team to collaborate in.

For a while, we were really working on trying to get the end-to-end vision realized and showing to companies that if they just adopt Meltano for their entire data lifecycle problem, they would get a tool that basically allowed our team to get those same advantages from things like code review, version control, continuous integration. But what we also realized is that at the time it might have been too early to try to get users or contributors on board with such a broad vision that requires a team, an entire team to adopt this platform all in one go. It wasn't very much built for kind of piecemeal adoption where you could start with Meltano just for data integration, and then maybe later on try out something else and convince your other team members to try to rest.

So one of the big lessons took place about a year ago, which is when we pivoted to focus specifically on the data integration part of the story. The realization we had is that while in the open source data tooling space, there exists really great solutions for transformation. I think I've mentioned DBT before. Orchestration, like Airflow, Dagster and Prefect. And for the visualization step as well, tools like Superset and Metabase, are getting very mature. But the answer on the data integration side of things, extract and load, getting data out of source systems into the data warehouse was lacking. Singer existed as a technology. That's the standard for open source data connectors. That was originally created by Stitch, one of these close sourced data integration vendors. But there wasn't a great solution or great answer to how to actually use those connectors in a fully open source environment that you can self-manage on top of your

own infrastructure, instead of these connectors only functioning essentially as plugins for Stitch's hosted framework.

So a year ago, when we realized we hadn't really been able to pick up the users and the contributors that we had been hoping to with the broad end-to-end vision, we started talking to some of the people that we had been able to attract and had been following the Meltano story since 2018. And we realized that the biggest demand, the biggest kind of vacuum in the data, open source data space that we had identified was specifically on that data integration side of things.

So really quickly, by not even modifying the product too much, but very much changing the messaging and the positioning on the website and the material that we were publishing, within a couple of days, usage numbers, contribution numbers, and community activity started picking up significantly. So depth speaks to the importance of positioning and marketing. We literally didn't change a line of code in the platform itself. But from one day to the next, people saw it differently, started using only one aspect of this broad platform instead of the whole thing. And since then, we've been able to attract people or kind of get them on board with the data integration solution that Meltano offers, and then get them to try out all of the other kind of tools that we integrate with that make up the data lifecycle, like DBT, for transformation, and having Meltano handle the airflow configuration. And we, in the near future, will also be adding integration with tools like Superset and Metabase, that you can manage the configuration of your entire data lifecycle products in one place so that you get those advantages of version control, different staging, and production environments, etc., that we think are very much the future of data tooling as data teams increasingly start thinking of themselves as software development teams that just happen to have a focus on data. And they're going to have the same kinds of expectations from their tooling that software developers have had over the last five years or so when it comes to continuous integration and platforms like GitLab.

[00:08:58] JM: The interview with Danielle a few years ago stood out to me for a number of reasons. One, it showed that the model of building software that GitLab pioneered. And make no mistake, this was a pioneering way to design a company. The idea that you would have this framework of – you basically take a SaaS category, as big as DevOps, and you say, "We're going to hang some scaffolding in this category. And we're going to kind of build around that

scaffolding. And we're not sure if we're doing it right. But we're going to iterate and we're going to make it better over time." It was a profound concept in company building. And GitLab has been richly rewarded for that. But the idea that you could do that for other categories was groundbreaking to me. So Meltano, it's no surprise to me that it makes sense in this regard.

But the other thing that stood out about that interview to me was this was the first time I heard about DBT. And Danielle was singing its praises. I was like, "I've never heard of this technology before that she's speaking about." Fast forward two and a half years later, DBT is profound company. Maybe as big a technology – I don't want to get too inflationary here, but maybe as big a technology for the data world as MapReduce maybe? Potentially? Saying too much?

[00:10:16] DM: No. I don't think that is overstating it. Like DBT is amazing. We're really very great friends with DBT. We have a really good relationship with the people at Fishtown. And what it has done for those not familiar is broads kind of data analysts who were tasked with putting together these transformation processes to pull data from different sources and combine them to create kind of the data that they wanted to use in their dashboard or that they wanted to use for particular kind of decision making outcomes. And take them away from these kind of points and click drag and drop tools in the browser that kind of follow the block-based programming style from tools like Logo. And it's introduced them to using SQL to define these transformations essentially by defining a SQL view over the incoming data using SQL define what you want the columns at the end of this stuff to look like. And then DBT orchestrating the execution of the SQL queries, making those tables concrete, and then having the data analysis visualization tooling point at the concrete tables that are the result of the SQL queries that are essentially the transformation steps.

Probably, if the DBT people are listening right now, they think I'll have butchered their pitch. But coming from a development perspective, that's kind of how I think about it. But what this has done is brought these data analysts who have been relatively non-technical and more used to kind of web-based applications and dragging and dropping and pointing and clicking into the software development world and showing them that by writing something that is code, but it's pretty easy to grasp, it's still SQL, they can start thinking of themselves as software developers and use things like code review, which immediately increases the quality of their work, because their team gets to have inputs, continuous integration and deployments that you can actually

verify that the SQL query is going to have the desired results before you actually throw it in production and your CFOs board meeting dashboard breaks. And we are seeing a shift of these people moving to a little bit more technical tooling and getting those same advantages that software developers have been getting from these kinds of technologies like Git for years now. And it's a massive shift, because it also allows the more technical data engineers on teams and written to less technical data analysts to talk with the same vocabulary and really get in sync and collaborate on the data project that they are team members on rather than treating themselves as kind of different disciplines that throw stuff over the fence, ask each other questions in phrasing they don't understand. And it makes a massive difference to just bring these people together.

And Meltano is very much a continuation of introducing more data professionals to software development style data tooling built on the strong conviction that data teams increasingly are going to be software development teams with a data focus rather than something totally separate that deserves totally separate tools.

[00:13:07] JM: Data engineering has basically been around forever, since the beginning of software, in some primitive form. I don't know the lineage too deeply since before MapReduce. For me, the world of modern data engineering begins with MapReduce. In any case, whether that's how you demarcate the beginning of data engineering or not, a lot of companies have legacy infrastructure in their data engineering pipelines. And if you have a data engineering pipeline with a bunch of legacy infrastructure, probably you don't want to stand up this brand new Meltano thing. Maybe you do. You could tell me if that's wrong. But I see Meltano is really well-suited for the typical startup where you start with something like a marketing automation company. You're building some marketing automation product. You're just trying to sell to marketers. You're not doing anything with data engineering. You're just trying to build some API or something that people are going to pay you some money for. And then you get some traction, and then you get more traction, and then you grow. And then all of a sudden one day you say, “Hey, we need data engineering.” In that situation, you can take Meltano off the shelf. Is that your target customer?

[00:14:19] DM: That's a good question. I can get a little bit more into the legacy side of it in a second. But one of the things that Meltano has definitely been built for is people with software

development background and expectations who are now getting data challenges, data problems that they are tasked with resolving. They will feel right at home because it's a command line interface, it has YAML files, it's a Git repo, etc. The other side of the fence, and we think that these kind of groups are going to merge over time, is these data engineers or data analytics engineers that have these less software developments that don't so much have a software development background and are newly exposed to software development tooling and are starting to figure out the benefits that they can bring to their teams.

So on the one you've got software engineers who are starting to tackle more data problems, and you've got data engineers who are starting to think of their problems in terms of software. And that means that we have two different audiences that we also have to kind of market ourselves to a little bit differently. But since I see those fields merging, that's just something which is kind of a vector of this particular stage in the data ops era rather than something that will be the case forever. But that does mean that if you're a tiny little startup, you have two engineers on board, maybe. And now suddenly, you have to set up your data integration pipeline and you have to start doing data transformation and get some dashboards spun up. That's combination of Meltano and DBT, and some of these other open source data projects that are kind of extensible, and you can go into the code and debug them if you're frustrated with things not working the way you want. It is pretty much perfect for that scenario.

When you are a larger team that already has a lot of data engineering expertise, they have all of these different tools spun up. They might have switched from one tool to another every couple years, as this space has been moving so fast. Then Meltano would just be yet another tool to introduce, and there might be some resistance to this. But it's still something I would recommend companies try, because one of the advantages Meltano can bring is offering a standardized solution of building these custom connectors for sources and destinations that might not be supported by your data integration solution of choice. Most data integration solutions today, some of the famous hosted ones with their billion-dollar valuations, their connector libraries really tap-out somewhere in the 100s to 50s, closer to 200 if you're lucky. But that doesn't begin to cover the entire breadth of SaaS tools and data formats that companies are using.

So today, if you want to pull data from a source that your data integration vendor doesn't support yet, you can either build a custom Python script and run it in Airflow. And you'll end up maintaining this indefinitely in-house. And it won't quite have features like incremental replication, or great monitoring and logging, or even something as basic as, “Well, this connector supports all of these different streams. But now I want to only pull data from this particular entity type.”

So an advantage of building your connectors to the Singer standard, instead of just writing a one-off Python code, is that you get a lot of this functionality out of the box. The Meltano SDK for Singer taps and targets that we have recently built actually makes it super easy to build a new connector and focus only on the source specific code where you can use requests Python library or some pythons client API library for the API in question. And the SDK framework then handles everything to do with adopting or complying to the Singer specification and allowing your little connector project to be plugged into any Singer compatible tool like Meltano. And because it is written to a standardized specification, it's something you can easily open source. And there will be thousands of people around the world who will be comfortable contributing to it, fixing bugs in it, instead of that being something you have to do in-house. And since it speaks the Singer protocol, tools like Meltano can immediately offer functionality like incremental replication, and better monitoring, and logging, and tracking of metrics, and allowing you to identify problems with your pipelines by then also integrating your pipelines with something like DBT tests for great expectations that you can learn about whether the data still has the expected format, which is much, much more difficult if it's just some Python codes one of your engineers wrote as a one-off.

[00:18:21] JM: So when I look at how this market is unfolding, I see a lot of importance in the orchestrator. It's similar to what happened in the Kubernetes world, where if you win orchestration, you kind of win everything. Not completely. But it's such a key component. And we're in this weird time where the dominant orchestrator is Airflow. Airflow kind of feels like the – I don't know if this analogy makes any sense to you. But it kind of feels like the Apache Mesos of data orchestration, where it's used people like it, it does the job, but it feels like there is room for something better. And that's the thesis of Prefect and Dagster. I guess it's also maybe perhaps the thesis of Astronomer, or I think there's maybe another Airflow company. But Astronomer, the Airflow company, if they can be the Mesossphere, or the D2iQ, that's not a bad

place to be if they're the D2iQ of data engineering. But I don't know. I'd love to get your perspective on that analogy and where you think the kind of data stack is going with regard to that hard dependency of the data orchestrator.

[00:19:39] DM: Yeah. So data orchestration, or in the case of these tools, kind of like workflow automation and their DAG processors, it's that problem of taking different steps in your data pipeline, tying them together, managing that, managing the schedules, managing the error handling, etc. I think Dagster and Prefect both have a really interesting kind of approach that is heavily inspired by Airflow. In some cases, they also have original Airflow developers and advisors to these projects. And I feel like, over the coming years, either of those or both of those are becoming the standard where Airflow has been that for the last couple years.

I think what you're seeing there's that Airflow very clearly was one of these kind of early open source data tools that now has kind of had its heyday, and with all of those lessons, and also adopting data principles more explicitly, there's a massive opportunity for Dagster and Prefect to take those lessons and just do it better. That does mean though that companies would need to migrate from one to the other. And that can require rewriting DAGs, or hoping that there's some way of converting DAGs from A to B. But it's kind of inevitable that that will happen.

I think the way that Meltano can fits in here is by saying, “We are not ourselves going to get into the orchestration or workflow scheduling problem. We will provide that foundation, that lowest layer of a team's data product that allows these different tools to integrate really well and have a consistent configuration layer and deployments methods.” So Meltano integrates with Airflow. And we have plans to integrate with Dagster and Prefect as well. One of the advantages this brings is that a team can start with Meltano with one particular set of plugins. But over time, as the needs of the team evolve, or as some new hot open source data tool arises, they can easily switch over to the new tool without having to completely rewrite the data integration part of their pipeline, the transformation part of their pipeline, because Meltano is able to kind of provide the payloads what happens inside these DAGs, while the orchestrator remains responsible for actually scheduling and running them.

So since in the data space, the data tooling is evolving so quickly, and every couple of years it seems like there's a completely new standard. And whatever's old is no longer cool, even

though two years ago it might have been drawing tens of thousands of people to conferences. It's a tool like Meltano that will make it much easier and more accessible for teams to migrate over to newer solutions, because they can kind of assume that if Meltano supports it, then I'll be able to adopt it easily and it will fit into my existing workflow. I won't have to completely have to learn again how to deploy or configure these things, because Meltano takes care of that integration between the different components in the lifecycle, which we are currently primarily thinking of as data integration, which is where Singer comes in, transformation that's DBT. Data orchestration, currently Airflow, Degster and Prefect to come, and then data visualization with tools like Superset and Metabase. And we are starting to also add support for data testing with tools like great expectations and DBT test, and for various tools to do with data lineage and observability.

I feel like Airflow is last generation data tooling. And we are very much at the start now of the next generation. And I think Dagster and Prefect have a really great position to become the dominant orchestration solution. But since right now, building a modern open source data stack for a team takes so many different tools and tying them together. Meltano has kind of a unique position underpinning a team's data project and bringing these best in class tools together. While we can be somewhat unopinionated with regards to who the winner is going to be, we just want to make it really easy for teams to use whatever they want to use and migrate over to the winner if they so choose.

[00:23:29] JM: That's very diplomatic. Do you have any perspective on who's going to be the winner?

[00:23:33] DM: Between Dagster and Prefect?

[00:23:35] JM: Sure.

[00:23:36] DM: I really like Prefect's model of kind of decoupling the actual execution of the workflows versus the scheduling, because I think, increasingly, and this is something that Meltano taps into too, there will be situations where teams would rather have the data not flow through systems other than those they manage themselves. This could be for privacy or security reasons. Or you might have to do with HIPAA compliance if you're working with health data, or it

could just be a simple matter of massive amounts of data and not wanting to pay for an event with and having that flow through some other provider.

So I think that Prefect is onto something by decoupling the scheduling. Making sure that the scheduling infrastructure never needs to see any data while the actual logic that you have written yourself that can run on your own infrastructure without having to deal with that scheduling problem. But I think it's far too early to pick a winner. Both of these projects have really great teams and communities around them. And I think there's going to be a good amount of kind of cross-pollination of ideas too as these teams and others figure out what's the next generation of orchestration tooling is going to look like. But Airflow increasingly is starting to look like a really great source for inspirations and things to learn from, and also things to do differently, because they haven't scaled or matured as well.

[00:24:59] JM: Have you met Nick? Have you talked to Nick from Dagster?

[00:25:02] DM: Not directly. No. But we're in some Slack workspaces together.

[00:25:06] JM: Okay. Well, maybe I'll introduce you. I think you would mutually benefit from a conversation.

[00:25:10] DM: Definitely.

[00:25:11] JM: Yeah. So what I like about Nick's approach, Nick is a cocreator of GraphQL, and he knows developer ergonomics in a way that few other people have explored. And that just puts a lot of like blind faith. I invested in his company, full disclosure, but I kind of have blind faith, just because I've seen him solve this kind of problem before. And it's a world where that kind of skill kind of trumps everything else, because, basically, it's a last mover advantage type market where you can have the best solution and be late to the party and still win. We saw that with container orchestration. Kubernetes was not by any means the first solution. There were like five other orchestrators at the time. And then Brendan Burns just said, “Okay, I figured it out. I did the best one. Here it is.” And that's what happened. Anyway, not to try to influence the direction of the orchestration wars. I talked to Prefect. They're doing noble work. Anyway –

[00:26:13] DM: Yeah. I mean, it's probably another coincidence that on our roadmap for more tools to integrate and support, there has been more demand than requests for Dagster than Prefect. Of course, like you said, it's way too early to tell just by usage numbers or something who's going to win. But I like that there is so much happening in the space right now. And both projects will benefit from both existing, and all users will benefit from competition in this space.

[00:26:37] JM: Let's take that customer type, like the typical customer type. Okay, I'll put it this way. I started a new company recently, or I'm working on it.

[00:26:45] DM: Congrats.

[00:26:45] JM: It'll be a company. Thank you. It's a game. A game has basically nothing to do with data engineering from day one. But there will come a time in the future where I'll say, “Hey, things are actually working.” All of a sudden, I want to do data engineering. If I was evaluating the market, my approach to data engineering might actually be to say, “The first thing I'm going to do is use Segment. And then I’m going to take Segment and I'm going to just put Segment all in my infrastructure. I don't care how much it cost. I'm just going to go all in on Segment, because it's the highest level form of data engineering.” Why would I do Meltano over Segment?

[00:27:22] DM: Well, especially if you're starting a company like this and you're going to have engineering people on board, but not anyone with data in their title. When it comes to setting up a data integration or data engineering solution, one thing you're going to look for is, “Well, we're developers. We expect this kind of stuff to live in a repository. We expect that changes need to go through code review. And we expect that the deployment will be handled by continuous integration and CD.” And ideally, it's something we could self-host. And if it's a Docker container, then I know how to do that.

Of course, you could also, if you have a little bit of exposure to data tooling, just find Segment. Go to their interface, pay them, blog, and point and click to put together two connectors and enter your credentials, and then just let them handle it for you. But if you're a developer, you will probably start feeling a little bit limited. And you'll start feeling the hints of inflexibility the moment you set up any kind of web-based SaaS tool. And I think Meltano is going to be a really compelling solution, because it is that easy to set up a data integration pipeline and deploy it

with the existing library of more than 200 connectors in the Singer ecosystem, and a super low barrier to creating new connectors. And I think developers who give this a try, even maybe before Segment or after Segment, will just fall in love with it really quickly, because it approaches data integration in terms and with tools that will just feel extremely natural to developers in general, software engineers in general.

So, yeah, if you want to spend no time at all setting up your own infrastructure or even worrying about, “Okay, I have to set up this repo and then deploy it somewhere,” then a tool like Segment is a really great option. But what we are seeing is that pretty much every company that starts with one of these hosted off the shelf data integration solutions with their heavily curated in-house library of connectors will eventually have a data source that is not supported by that platform. And then as I described earlier, you end up writing your own code anyway. And then you start thinking like, “Well, now I have two data integration solutions.” Or maybe you’re one of the companies that actually becomes a customer of multiple closed source hosted data integration solutions, because they have overlapping libraries, and you want some of the one and some of the other.

And then you'll start thinking, “I wish I just had one standardized approach that fit into my development workflow where I know that there will never be a limit on connectors, because it's super easy to write my own. And there are hundreds and hundreds already being maintained by the community.” So I feel like as developers are more inclined in general to just go with kind of codebase tooling that they can deploy in a Docker container, Meltano will be the absolute best choice for a software development team that suddenly has to deal with data.

[00:29:57] JM: So what's the onboarding like? So let's say I’ve got likes some interesting customer transactions that are happening across my game. For example, like let's say I have a game that has some purchases, or microtransactions and stuff like that, and I'm like, “Okay, I'm finally at the point where I have some product market fit. I've got some money coming in the door. Now I really want to start studying my data. And I want to see what is leading to profitability.” And let's say like my data is all in Firebase, or Firebase and maybe BigQuery, or in some blob storage, or standard places that I'm throwing this data. What am I doing to get started? What's the road from Meltano day one to actual productivity?

[00:30:40] DM: Yeah, that's a good question. So the first thing you would do is install the Meltano PyPI package, pip package on your local machine. Just like most software developments, you start in your development environment, and then the step of deploying to production comes later. But what's going to be running locally is going to be pretty much identical to what you would end up deploying. So you install the Meltano package, then you create a Meltano project by running Meltano in it with some project name, which builds a product directory with a Meltano.yaml file that will hold the configuration for the different plugins that you're going to bring into your project.

The next step then is to add a couple plugins. And in this case, the plugins you're looking for are first of all an extractor, or a Singer tap for, like you mentioned, Firebase, BigQuery. Maybe you've got some data in Postgres. All of these have well-maintained connectors that are not all supported out of the books by Meltano, but through the Meltano hub for Singer taps that we have set up. It's really easy to discover these and find the installation instructions.

So then you run a couple of commands. And then you end up with this definition for tap Firebase in your Meltano.yaml file. And you can configure it using the Meltano config CLI. And then the next step is to pick a loader, which is a connector for a destination. So this depends on where are you going to want to have all of this data end up. If you already have BigQuery set up with some data that's coming from your game or from your customer transaction handling infrastructure, then BigQuery is a natural choice for your data destination as well. So then you would install the target BigQuery loader. Similarly, configure it using the Meltano config CLI, or by directly modifying meltano.yaml. And then you can immediately use the Meltano ELT command to take a tap and the target and run them together so that any data that the tap manages to extract will be directly fed into the loader in order to be loaded into BigQuery. You will do this three times if you have three different sources, but you can use the same loader configuration each time. And then you end up with three different pipelines. You can run these locally just by running the CLI command. But if you want to automate running these pipelines, you can additionally add Airflow into your Meltano project. So you would run Meltano, add orchestrator Airflow. Meltano would add the Airflow pip package into Meltano's own definition, meltano.yaml, and you would get the opportunity to configure airflow. Although the default configuration should be enough to get you started, because Meltano also automatically builds a DAG generator, an Airflow compatible DAG generator, that looks at Meltano's scheduled

pipelines and creates a scheduled DAG for each of them. So then the last step to get this running locally is to actually start Airflow scheduler using Meltano invoke Airflow scheduler. And then you have pipelines running on the schedule you configured with the credentials you've configured automatically pulling data out of Firebase and BigQuery and loading them into Firebase, or loading them into BigQuery, or whatever destination you had chosen.

At this point, you have all of this running locally in a directory that you can easily initialize a git repo inside, push it up to GitHub, or GitLab, or what have you. And then the next step is going to be how do I deploy this. In order to deploy a Meltano project with all of the plugins it contains, which in this case are the extractor and the loader and Airflow, you can really easily containerize your Meltano project by adding the Docker files to your project. You can actually run Meltano add files Docker, and it will immediately add a Meltano specific Docker file, a Docker ignore file, and everything else you would want to build a Docker image for your project. That then exposes to Meltano's CLI SD entry point so that you can host this Docker container somewhere, tell it to run with the Meltano invoke Airflow scheduler commands. And then you will have this environment running that will automatically run those pipelines you configured on a schedule.

So you can do this locally with your local Docker, of course, or you can do this in production if you want to build the image and push it somewhere. But if you want to use continuous integration and deployment on GitLab CI or GitHub Actions or something else to do this, you can really easily add the appropriate GitLab CI configuration file to your Meltano project as well by running Meltano add files GitLab CI, which will add a GitLab CI YAML file that will automatically build the Docker image using that Docker file that have been inserted at the previous step. And then has kind of a deployment step where you can build an image, push it to Docker Hub, or to GitLab's own registry. And from there, you can point your Kubernetes infrastructure or whatever Docker runner you've got at that Docker image and tell it like I mentioned before to initialize it with that Meltano invoke Airflow scheduler command.

So if you're a developer, all of this sounds pretty straightforward. You've got a CLI, you've got a Git repo, you've got a Docker image that gets built. And then you just have to initialize the Docker image with the appropriate commands, and then your whole data integration setup is going to be running.

[00:35:39] JM: Wonderful. And the actual time it takes to do this? What would your estimation be? Like are we talking a week? Two weeks?

[00:35:48] DM: Of course, that's kind of depends a little bit based on whether the connectors in question already exist for the source that you want to integrate, or whether it's well-maintained enough that you can just use it without hitting any hurdles. Because in some cases, these connectors have been built by one particular data team, and it works in their scenario. But then if you're using it with a different version of the source or something, or if the API has changed over time, you might run into a Python error that you might have to fix this yourself. But that is just the nature of open source. And you'll see the same with API client libraries. So that might take a little bit extra work if you want to fix that bug. But if you're a software engineer, that should be relatively straightforward.

Otherwise, all of those CLI commands I'm talking about, if you know ahead of time what you're planning to do and you know how Meltano works and how it expects you to set up a project, you can get to that final running and production step within an hour, no problem. I recorded a speed run video a couple of weeks ago of going from zero to running a data integration pipeline that pulls data out of GitLab and loads it into a Postgres database, and the entire video take 90 seconds, because of course, that's me knowing exactly what to type. But if it takes you weeks to set up a project like this, then we have failed in our mission, because we have aimed to make it extremely easy to put together these data pipelines and to deploy them. And if you are comfortable already with deploying Docker containers on to Kubernetes or similar platforms, then everything on the development side of things on your local machine should be done within 30 minutes, and then getting it running somewhere externally in production should take another 30 at most.

[00:37:18] JM: Just to press you on this point –

[00:37:20] DM: Please.

[00:37:21] JM: Like time to market with Segment, I'm putting in a few lines of code to start spitting metrics into Segment's big set of well-defined user experience infrastructure. Meltano, I'm getting improved configurability, but I'm not getting this real ease of use streamlined just

insert a little bit of API code Stripe-like experience. I mean, are you sure you're not just catering to kind of a different price point type of customer? Different abstraction level of customer? Are these actually comparable product experiences that I'm making here? Or am I totally off-base?

[00:38:00] DM: I think there's a significant difference there, which you kind of called out by saying, “With Segment, I just have to add a couple of lines to my code. And then we'll end up sending this data to Segment.” The approach that Meltano takes right now, which is pretty typical for these kinds of data integration solutions, is that it has a connector that connects with a data source that already stores the data in some form and exposes it either through a SaaS API or a file somewhere in an S3 container. Or it's a database with a well-defined protocol for pulling data out of it.

So Meltano assumes that you have already somehow gotten your data into Firebase or into one of these data stores. And then Meltano can directly connect with that source and pull the data out and then load it into some other destination. If you want to add little bits of lines of code in your existing game implementation and all of the places where you want to push some specific bit of data to Segment, Meltano doesn't help with that currently. It does not have its embedded data store that you can write some lines of code to push data into. It assumes that you've already written the lines of code that get the data somewhere. And then Meltano can just put it out there and go from there.

So Segment and Meltano, I would say, are not directly comparable because of that distinction. Meltano is more comparable to tools like Fivetran and Stitch that have this list of connectors for sources and destinations and allow you to configure those and set up your scheduled pipelines. But we are working with people in the community and seeing if there are other open source tools that we can adopt like Snowplow and hook those into your Meltano project that Meltano can directly take the data that your Snowplow snippet might have sent just like Segment, and then move that to the destination where you want it to end up.

[00:39:43] JM: Switching topics completely. What's the mechanism for spinning out a company these days? If you want to spin a company out to GitLab, how do you appropriately do that equity-wise, ownership-wise, venture capital-wise? What's the best strategy?

[00:40:01] DM: That's a great question. I think the Meltano spinout story is a little bit different from some other stories you might have read about open source projects spinning out of the organizations in which they were created with a new entity set up around them. One significant difference is in the fact that's, from the beginning, back in 2018, the goal for GitLab with Meltano has been to build a really compelling tool primarily initially for its own data team to benefit from, and then to kind of turn into a separate product line or business unit within the GitLab organization. So it's not one of those stories where just one of the people on the data team at Airbnb or Netflix built something really cool, and then just kind of ran off with it. It specifically has been incubated inside GitLab for a number of years, because we realized there will be a significant opportunity here.

So the way that the spinout worked is that in order to allow Meltano to really kind of spread its wings, we identified that it wouldn't make much sense for it to stick around within GitLab, because GitLab has been focusing increasingly and all the way from the beginning really on this, the single application for the entire DevOps lifecycle. And we didn't see a path to fitting Meltano into that bigger platform. It was basically always going to be a separate project.

And at the same time, we were realizing that the kind of infrastructure and incentives around compensation that are set up in a big company like GitLab with 1,300 employees right now, are not quite what you want if you're starting to put together an early startup team that is going to try to get a brand new project off the ground.

What all of this means is that, in the spinout, GitLab transfers all of intellectual property into the new entity and receive shares in return. And then we started pitching to venture capitalist firms that we have been building relationships with over the last year to see if any of them would bite in order to invest in the seed round. We got a couple of term sheets. We chose one party that we were really eager to work with and just really inspired by their enthusiasm and also the kind of the investments thesis that they had around open source data integration tooling. And we attracted a number of angel investors to join the round. And now we've spun out. So GitLab retains a significant share, because it was incubated there, and it was always intended to be a business unit that would eventually bring GitLab additional value. And we've brought some investors on board to fund this next stage of our growth.

[00:42:21] JM: So where is data engineering going?

[00:42:26] DM: Well, my strong conviction, especially coming from a strong software engineering and open source background. For context, I've been at GitLab for six years now. I joined when the company itself only had 10 people working for it. But we already had an open source community, I think, in the many hundreds at the time. When I first started learning about the state of data tooling, data engineering tooling, data analytics, and analytics engineering tooling, I was just surprised to see that it was very reminiscent of the software engineering space five plus years ago before DevOps version control, code review, CI/CD really became kind of staples in every company that wants to build an efficient software engineering team building high-quality products and with high confidence that what they're building isn't riddled with bugs. I was surprised to see that the data space seemed behind.

So I very strongly believe that that space has been deprived a little bit of the advantages that have come to the software development space over those past years. And I think we are at the beginning of this new wave of data ops tooling that embraces DevOps principles from the foundation in its DNA, and really starts giving data teams the tooling they deserve to build products with more collaboration and more efficiency and just higher quality results At the end, leading to higher quality decisions, which is, of course, what companies ultimately care about when it comes to their data team.

Quick shout out to a talk that one of my Meltano colleagues, Taylor Murphy, did with Emily Schario, who was one of GitLab's – Was at GitLab for a while. And it's now data leads, or I'm probably messing up the title, but data something at Netlify. They did a great talk a couple weeks ago about how to run your data team like a product team. How to think of yourself as a team building a data product with all of your colleagues within the company essentially being your customers and your users. And this fits very much into this vision that we are chasing after with Meltano, where the data team is just a product team building a software product that happens to have their colleagues as the primary users, but they should approach it and get the same advantages of software developments' best practices that their actual product development colleagues have. And I feel like I think you'll start seeing more and more that people will expect tools that embrace kind of pipelines as code. Or in general, bring these data

challenges into the realm of code and get repos and stuff that we can discuss on a merge request or a pull request discussion page.

[00:45:03] JM: What do you think of the divergence in the Snowflake-based world versus the Databricks-based world?

[00:45:15] DM: How would you characterize that divergence, Jeff?

[00:45:18] JM: I would say that I see these two technologies as basically the open source data engineering platform company with the most potential versus the closed source data engineering company with the most platform potential.

[00:45:36] DM: Yeah. I think that, in this comparison, of course, Snowflake being a closed source version, and Databricks and Spark being an open source, I think that how quickly Snowflake has been able to find adoption speaks to the accessibility problem of open source solutions and the higher barrier to entry not just to like start it up and deploy it somewhere, but also to wrap your head around it. Like Snowflake is extremely easy to get started with in part because it is a hosted platform where you don't have to worry about any of that to yourself. But I think there is a massive opportunity for perhaps the next generation of open source data warehousing tooling, whether that's something Databricks comes up with in an iteration of Spark consumer technology, or something totally new, to have an answer to Snowflake. And I think that Meltano can actually help there with making the configuration and deployment and integration story really easy. We have started looking into open source data warehouse technology already that is similarly a kind of object storage-backed like Snowflake, and can really easily be brought into an existing Meltano project. And that barrier to entry, we want it to be so low that you can add one of these technologies by running that similar Meltano add storage full command. And having Meltano automatically hook it up with your DBT, with your connector, so that you can be off to the races.

We've started working with some community members on Athena supports in Meltano, which is also supported in DBT already, which I think is a great sign of things to come. But in general, I think that the data engineering space will be longing for that additional flexibility and debugability and extensibility that only open source software can offer. And then we are seeing that

increasingly data teams using closed sourced products are supplementing that with open source stuff on the site. And then it's just a matter of time before they move over to open source entirely.

[00:47:29] JM: If you're trying to say that if you look at Snowflake, and Snowflake is the closed source state of the art data warehouse, and we're now waiting for the open source answer to snowflake? Isn't that a little bit like saying we're waiting for the open source answer to AWS? Because Snowflake is basically so hard to build and so hard to orchestrate with storage tiers and all the stuff that you need to give this seamless experience. Like you could never operate an open source AWS. So why would you be able to operate an open source Snowflake?

[00:48:05] DM: I mean, I think there is a significant difference there whether you're running a massive platform with millions of users on it, versus something that can scale enough for one organization. And of course, open source kind of implies that you will be self-managing it or at least you'll be still hosting us somewhere in cloud infrastructure, but you only be responsible for scaling it to the extent that your organization needs. So I think the problem of building an AWS or building a Snowflake when you want to offer it to everyone is harder than building something even for one specific very large organization. But I don't have a great response to that question. I'm also kind of glad that that's not our problem to figure out, because the integration problem that we are focusing on, I don't want to say it's easier. The value we bring will be massive if we can introduce more people to the modern open source data stack. But I myself not in the business of data warehousing or data warehouse tooling.

[00:49:04] JM: Alright. Listen, I really enjoyed talking to you. And I think Meltano is a really, really cool project with a lot of potential. So, congratulations.

[00:49:11] DM: Thank you so much, Jeff. Glad to have you on board as a fan. And just one last notice, mention for everyone listening today, check us out on meltano.com. Join our slack community with upwards of a thousand members. And yeah, see what Meltano can offer for data integration and the data ops era to come.

[00:49:28] JM: And apply to join the company if you want to be at the next GitLab.

[00:49:31] DM: Definitely, definitely.

[00:49:33] JM: Alright, great talking to you.

[00:49:34] DM: Thanks so much, Jeff.

[END]