EPISODE 747

[INTRODUCTION]

[00:00:00] JM: Netflix has petabytes of data and thousands of workloads running across that data every day. These workloads generate movie recommendations for users. Create dashboards for data analyst to study and reshape data in ETL jobs to make it more accessible across the organization. Over the last 10 years, data engineering has become a key component of what makes Netflix successful. There are many different engineering roles who interact with the data infrastructure at Netflix. These include data analysts, machine learning scientists, analytics engineers and software engineers. Data engineering at Netflix has come a long way from the days of Hadoop MapReduce jobs running nightly and generating reports of the most popular movies.

As data engineering and data science has grown, the tooling has expanded. The people in different data roles at Netflix might use Apache Spark, or Presto, or Python, or Scala, or SQL, or many other applications to study data. But in recent years, there is one tool that has stood out for its ability to be distinctly useful - Jupyter Notebooks.

A Jupyter Notebook lets a user create and share documents that contain live code, visualizations, documentation and many other types of components. In some ways, it is like a sharable interactive development environment. It allows other people to see how you're working with your code and why you're making certain decisions and it's also a tool for building interactive user-friendly applications. You can embed videos and images in a Jupyter Noteboook.

A Jupyter Notebook stores both the code and the results of an application together in one place, and by combining code with the results in one document, you're going to have context around why a certain result came out the way that it did.

Matthew Seal is today's guest. He's a senior software engineer at Netflix, where he builds infrastructure and internal tools around Jupyter Notebooks. Matthew joins the show to explain what problems Jupyter Notebooks are solving for Netflix and why Jupyter Notebooks have quickly grown in popularity within the company.

We are gathering feedback for our 2019 listener survey and you can go to softwareengineeringdaily.com/survey to tell us how we can improve. We read every piece of feedback every survey, because we want to know how to adjust the direction of the show. That's softwareengineeringdaily.com/survey, and we're also looking for sponsors for 2019. If you're interested in reaching the 25,000 daily listeners or the 50,000+ subscribers to the show, you can check out softwareengineeringdaily.com/sponsor to learn more, and we could use your help. You can help us out by sending an email to your marketing director, or your CMO, or your CEO.

Many people are actually surprised by how effective podcast advertising can be to get their message out to technologists and it's often difficult for us to convince advertisers to buy podcast ads directly, but if they know that developers in their organization are actually listening to the show, it can be really helpful to us and we invest heavily back into Software Engineering Daily. You can rest assure that your sponsorship dollars are going to words growing our product offerings and delivering you improved content. As always, you can send me an email, jeff@softwareengineeringdaily.com.

We are scoping out what we're planning to do for this year. So if you have show suggestions, I would love to hear them, jeff@softwareengineeringdaily.com, and let's get on with the show.

[SPONSOR MESSAGE]

[00:03:59] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/ HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW]

[00:05:22] JM: Matthew Seal, you are a senior software engineer at Netflix. Welcome to Software Engineering Daily.

[00:05:27] MS: Thank you. I'm glad to be here.

[00:05:28] JM: Today we're talking about Netflix and data. Netflix has petabytes of data. There are lots of jobs running over that data, lots of engineers working on applications on top of that data. Give us a brief overview of the data infrastructure at Netflix.

[00:05:44] MS: Yeah so the data infrastructure at Netflix is it's very Amazon focused. So the whole Stack is sitting on top of Amazon servers, and one of the interesting things that Netflix did pretty early in the process before other people kind of jumped on this was they stored all of their distributed data in S3 rather than HTFS or some of the other systems connected to Hadoop and this gave them the ability to kind of upgrade and distribute and scale on top of Amazon's infrastructure instead of having them manage it themselves.

The system has created a lot of swirl around how to think about big data and distributed. In particular, I think a lot of the systems where you can have independent systems develop without being too integrated has been a pretty big advantage working there.

In terms of other technologies, we use basically everything under the sun, but I would say like Spark is probably the biggest growing and most commonly used system and with Pig being kind of the traditional tool used.

[00:06:37] JM: There are a number of different roles that involve data at Netflix. There's data analyst, machine learning scientist, analytics engineer. How did you get to this place where there's this large taxonomy of roles and how do they differ from one another?

[00:06:53] MS: Yeah. The large taxonomy of role is something you sort of see as the company grows. You start off trying answer particular business questions or solve particular problems, and the tools kind of follow what the business need is. So in there in the sense the Netflix started originally much more as just as serving other content systems on to Netflix, things like pulling in other producers, data or trying to match the DVD collection to the online collection, which before my time.

What happens is, is as the company grows and has been making their own original content or they've been trying to figure out how to continue being competitive in the market, these other roles popup and where you need to answer specific questions in order to determine what the best choice is. So things like analyst show up, because you start finding how you do your marketing for original content or how you do decisions around what the show users or what people are interested in or even regional differences like that's a really big deal is how a particular region or an area of the world is going to respond to different content.

So that requires being able to answer questions about how people are responding to what they're seeing and how they're responding what they're doing, which requires more and more different types of analysis and data collection, which then dictates to how more data to parse through, which means more data engineers to support those people and it solely becomes this like hierarchy of people all support this like top-level business need.

[00:08:16] JM: You can wind up with a lot of different tools with all of these different roles, and it can be difficult to know best practices for what tools to use. How do the tools vary across the different roles at Netflix?

[00:08:32] MS: Yeah, that's actually a really interesting question, and in particular at Netflix, there's a lot of like smaller like groups of people who are all working kind of towards their own goal and then try not to dictate everyone use the same exact tools, because we know different people in different circumstances are going to be more successful using tools more specialized to what they're trying to achieve.

What results in is that there being a lot of different tools, and it's a good and a bad problem. Actually, a lot of the work I've been doing is helping for some of the really common use case problems, where there's a very clear pattern of how we should approach a particular problem, like orchestrating ETL, for example. In those spaces, a lot of people are solving the same problem with different tools. We're trying to consolidate a few tools there without turning off all the tools available for people who need more specialization.

I would say the tool list that we have is pretty expensive. We use everything from Tableau, to all the different big data systems, like Druid, in Spark and all of that, but there're a lot of different orchestration and CI tools that are all omnipresent.

[00:09:36] JM: With all of these different roles and all of these different tools you can wind up with redundant work. You can have a data scientist do some exploratory analysis and then that exploratory analysis, when doing that, they might write some script that would be super useful to somebody in another area of the company, but collaboration is really hard. How do the different data people collaborate with each other across the organization?

[00:10:08] MS: Yeah, actually it's not just collaboration. Potentially, people focus on the data that's also collaboration on to tooling as you mentioned other things. I would say in general it's something that Netflix is probably trying to solve better. I mean it's one of the value props with adapting notebooks heavily is making that more possible. I could talk about that more later.

I would say today, generally, within a team or an org, each of those orgs has their own system for how to understand what's there. For the big data platform, in particular, we've been really emphasizing trying to make this big data portal our centralized landing location. It has the ability to search and identify useful code and data.

When you're looking for answering a problem, we want to make it so that you go there to find out what you should do and where you should look and to find examples of what else is being done. Today, most of the sharing is done by tem knowing kind of what adjacent teams are working on and making sure that people meet up on a periodic basis to understand what other teams are doing so that you don't have as much replication of work.

[00:11:14] JM: Let's talk about notebooks. What is a Jupyter Notebook?

[00:11:18] MS: A Jupyter Notebook is in essence is just a JSON document, which represents code snippets in these delineations called cells as well as their outcomes, or they call their logs, or outputs that they generated coupled all in the same document. That gets a little bit confused with like the notebook interfaces. So the notebook document is just a very simple way to store what code is being executed and how it's actually executed and it's meant to be in a way that's shareable across many different platforms.

The notebook service itself, something that host a notebook and renders it, is basically just a frontend to serve that document and follow the protocols and specs that the notebook requires in order to operate. At the end of the day, what it really means is this is just mostly like describing an interactive coding environment that can be hosted and to have a record of how that was executing.

[00:12:10] JM: Notebooks were popular for I think a longer period of time in academic research before they started getting increasingly used at companies. Can you describe a little bit of the history and the different use cases for a notebook?

[00:12:25] MS: Yeah, definitely. I think the history is pretty rich. I mean, I was even using them back when I think Jupyter just came up. I started using them for doing talks and demos of what I was working on. But overall, the history of Notebook kind of goes really far back into spaces where you had kind of a branch in how IDEs are developed where you want to do iterative coding. In particular, where you have long-running processes that kind of are expensive to compute, but you want kind of a repel that's sitting there that's running, but with code charts that you could rewrite on-demand.

So the use case of this was really attractive for machine learning mainly because in machine learning you end up often times doing very expensive either ETL data collection or data cleansing, and then you're iterating on some small piece that's cycling on how to analyze that data. Once in a while, you need to run the code further up the stack, but you don't always want to rerun everything, because that adds a lot of time to you iteration cycle.

The traditional use case was kind of when you had these exploratory or scratch pad or kind of iterative cycle where you didn't want to repeat all the work. It was a really good form factor for following that type of code execution. What happened since then is that the tool is actually being used more and more for both reporting and for kind of doing a more deeper dive on how you would solve something. But this ends up creating friction when you do a deeper dive using a notebook, because the notebook you end up putting all these code into then gets retranslated out into another program which requires you to kind of rewrite what you did.

So one of the things that's been changing recently is if you've written a well-written notebook and you have it operate the way you expect, we actually have started to shift at Netflix to actually just schedule that notebook. So you can actually run the notebook as you prescribed it without having to report it to another form factor.

[00:14:16] JM: For people who are unfamiliar with the topic of a notebook, it might be a little bit unfamiliar. So I think I want to explain it a little bit more. How would you compare the usage of a notebook by a data scientist or a data engineer or a data analyst that notebook usage? How would that compare to how a software engineer, like an infrastructure engineer would use an IDE?

[00:14:41] MS: Yeah. I think the place where this compares is any place where you want to have code where you're rerunning it. So like complex IDEs tend to have the ability to run a particular unit test or debug this particular section of code or rerun this subprogram that's already been specified. The notebook is kind of tailored somewhat to that type of use case experience, where because you have code broken up in these cells and they each log independently what's happening. You can run the cells, like you tend to group cells, like all my imports are in one cell and then I'll have some constants in another cell and then I'll have maybe a couple of functions — functions I have defined that I want to have here that aren't necessarily important enough to put into a library.

Then I'll have kind of logical pieces of how my program is going to execute. A lot of the traditional use of notebooks has been how you think of like a script. So maybe in an IDE if I were developing like a Pi Script where I just want to execute something that I could call on-demand, that's a pretty common corollary, but here instead you've broken the script up in some discreet pieces so that you can manage them independently.

Notebooks have taken an increasingly important role within Netflix, and I was pretty amazed in reading your posts and watching some of your videos about just how deeply notebooks have taken over Netflix and how you've changed the infrastructure to support the usage of them.

[00:16:07] JM: For an engineer at Netflix, for let's say a machine learning researcher, a data scientist, one of these data roles that are using a notebook, what have the notebooks replaced? What were they using before that?

[00:16:19] MS: Yeah. I think it's an interesting question. So to give a little bit of context, so at Netflix as you've mentioned, it's not just like a few users using this. I think it's something like 10 or 12% of all Netflix employees have edited a notebook in the past month. The use case – Where they were used before for tooling was it was a mix of things. It was either very unorganized. It was something like I have just some scripts that I have somewhere in S3 and I've put them there and then I forget about them, and the next person comes along, has to kind of rewrite stuff, because they don't know where or how I organize my code.

[00:16:54] JM: Scripts, just like Python scripts or batch scripts or something.

[00:16:56] MS: Yeah, R scripts, Python scripts, batch scripts. Yes, that happened a lot, and you also have a lot of like writing code that interact with a tool and then you would try and build it into the tool as much as you could, but then when you move tools, you would kind of have to rewrite everything. So I think it's been kind of a slog to make it happen and get there to just make your machine learning model work.

Now, that's the traditional use case for notebooks. I think the other side with the like analysts side of things is actually that the role is such a new and growing role that we didn't have very many tools or solutions for these people, because we didn't have enough investment in making their lives easier. So I think for a lot of them, they've been relying on having not necessarily a tool do these things for them, but another engineer.

So you have an or a machine learner that leans very heavily on data engineers to translate something they've sort of hacked together in a script on their computer or some place or in some system that we've built, and then they have that data engineer really help rewrite the whole thing. Whereas here, I think you're getting a lot more self-service, where people are writing into a notebook that they can run and track and share easier. Then they don't actually have to lean on another engineer to transcribe whatever they've generated in different places into something that could be reused and shared.

[00:18:12] JM: When you decided to make – I guess Netflix decided to make a big bet on notebooks, I've heard you used that term a number of times, big bet. What did that entail when there was a decision to double down on the notebook usage and you say, "Okay, we're going to build a lot of infrastructure or supporting tools around notebooks, because these things are so useful." What did that big bet entail?

[00:18:36] MS: Yeah. Actually, there was a memo that went around Netflix, and the memo is labeled 'notebooks everywhere', and it was sort of an ode to this is a tool that's being used rapidly by the company. It's something that's being adapted very heavily outside of maybe the traditional engineers, and it was sort of a why not use this for other things?

A lot of the reasons that we kind of listed there for why people haven't used notebooks traditionally were things that, "Hey, these are all solvable problems," and they actually have a lot of value in the composition of a notebook. It's basically opening up the idea. There's a few cases where the notebook can be used with relatively low-risk to your normal day-to-day operating procedures, and you can kind of start to ingrain it into other use cases and then start building the tooling and the support for it.

So the big project that really kind of jumpstarted from a small team working on notebooks to many more people working on notebooks was the scheduler project, which I kind of landed on back about a year ago, and that project was basically to help consolidate our orchestration and scheduling projects for you to have an ETL script or report that you need to run on some cadence or after some data that's ready. The tooling we have for that, we have a lot of tooling for that and a lot of the tooling replicates what it's doing.

So one of the bets we were doing was moving to a single scheduler for the majority of this type of patch work, and it was built entirely off of notebook templates. Even though users may not know that they're running a notebook, rather than describing templates within code-nested deeply in some repo, we actually wanted to expose what we were doing and how it was operating much more forward-facing to the users.

So that means is the like 40 or so templates that we supply for like common patterns of execution you want to do in the platform. They're all notebook-based. When the user actually runs it, they kind of specify their parameters and pass everything through. At the end, the outcome is actually a notebook. So if you want to see how it ran or what it ran. They can always go click on their notebook link and go even replicate exactly how it executed.

[SPONSOR MESSAGE]

[00:20:49] JM: OpenShift is a Kubernetes platform from Red Hat. OpenShift takes the Kubernetes container orchestration system and adds features that let you build software more quickly. OpenShift includes service discovery, CI/CD built-in monitoring and health management, and scalability. With OpenShift, you can avoid being locked into any of the particular large cloud providers. You can move your workloads easily between public and private cloud infrastructure as well as your own on-prim hardware.

OpenShift from Red Hat gives you Kubernetes without the complication. Security, log management, container networking, configuration management, you can focus on your application instead of complex Kubernetes issues.

OpenShift is open source technology built to enable everyone to launch their big ideas. Whether you're an engineer at a large enterprise, or a developer getting your startup off the ground, you can check out OpenShift from Red Hat by going to softwareengineeringdaily.com/redhat. That's softwareengineeringdaily.com/redhat.

I remember the earliest shows I did about Kubernetes and trying to understand its potential and what it was for, and I remember people saying that this is a platform for building platforms. So Kubernetes was not meant to be used from raw Kubernetes to have a platform as a service. It was meant as a lower level infrastructure piece to build platforms as a service on top of, which is why OpenShift came into manifestation.

So you could check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[00:22:58] JM: Before we get into the discussion of scheduling and some of the detailed infrastructure that you've built around notebooks at Netflix, I want to focus a little bit more on the basics of notebooks, because I think there're going to be people who are listening and are thinking, "Okay, this is sort of like an IDE where I can write scripts, where I can execute my code." I don't understand why this solves all these problems around collaboration, or simplifying the tooling. Can you help us clarify why are notebooks so useful? What is the big differentiator versus other ways of developing software?

[00:23:42] MS: Yeah, I think probably the two biggest differentiators are going to be the fact that you can have visual representation of what you're executing. You can generate graphs. You can generate much like if you've used some of development tools like MATLAB or RStudio or other systems like that, you could basically visualize and log and see what's happening right next the code that's executing without changing your context or switching context or having to know as much about what code generated what outcome, and you get nice visual representation that's a little more friendly to maybe the less traditional developer crowd.

Then the other advantage is the fact that this code is broken apart into different cells. That may seem like a really trivial change, like can have different function to your own. That's true, but the way it gets represented is that you actually have this code that's like both physically on your screen and logically isolated, but is re-runnable. The emphasis of where you can kind of focus on what you're developing on a linear-ish fashion where you're kind of going through. Then when you find a problem backing up and fixing something without having to necessarily jump around and find everywhere where you need to make code at it.

I would say that it's not a tool that replaces all development or can replace all the IDEs or anything like that, and I think that's been a little misconstrued in a few conversations I've had, where there's still a very clear home for IDEs. I think notebooks are going to get better and better until there's less differences between lightweight IDEs and notebooks. But I do think for the use case where you want – Users want a guided experience to developing, but don't want the very heavy feeling of an IDE or the very like integrated have to know everything that's going on around it experience with maybe some of the lighter weight editors. It gives a kind of home for those people that are in between those different skill groups or those different use cases.

[00:25:34] JM: All right. Tell me if I have this right. So for the last I would say two and a half years doing this show, I've talked to a lot of different people about the frictions of developing a machine learning model or developing some complex data operation locally and then trying to take that to production and all the frictions that arise from that. Just as an example, maybe you'll have a sample dataset that you keep on your local machine and you use that sample dataset to do some exploratory analysis. You want to look at charts and graphs. Maybe you're using a BI tool to do that and then you're tabbing between your BI tool and your IDE where you're writing a script to do some data cleaning and to do some data analysis and you're hoping that you'll be able to extrapolate the results that you're getting on your local machine to whatever this would look like in production when it's on a huge distributed computing job. Then in reality, there's a lot of difficulties and even once you get it working on your local machine, getting it to production, the fact that maybe you're writing in one language on your local machine and then you hand it off to a data engineer. They have to rewrite it. Maybe they're using a different kind computer or they're using a different kind of processor or just the dataset is much bigger and perhaps some of the data is unclean. So testing it, there's an emphasis on you want to be able to test this stuff efficiently. I think notebooks solves a problem there.

But basically the idea is that any friction that we can remove from the inconsistencies of the environment in which this is running in sort of development versus running in production versus any of the phases between development and production, you want to eliminate those frictions as much as possible because this is difficult enough as it is. What Netflix saw was that notebooks are something that seems to be an interface that we can use consistently through all of these different processes.

[00:27:41] MS: Yeah, for sure. I mean, I think that hits it pretty well in the head. Actually, my background, I actually started on the machine learning side of things. So I intimately know the frustrations of trying to get your code to production in a way that's going to scale or even just run. For sure, yeah, that's definitely been an emphasis on trying to make it so that there's fewer points of friction so people – I think the term that gets used a lot in Netflix is actually self-service. We want to make these systems both machine learning and analysis more self-service oriented. A user should be able to describe what they want to do, do it, and be able to productionize it with as minimal support from other people as possible.

[00:28:18] JM: So what's the process of developing my notebook locally and then getting it into production? I guess maybe you could give an example of a situation when we would want that to happen.

[00:28:32] MS: Yeah, for sure. Actually, there's a few ways that you can think about it from local development. One is you could be developing actually on your machine and you've launched like a Jupyter server or a Jupyter lab or one of the other frameworks out there to run a notebook, and you could launch that yourself and then develop the notebook locally and then you could actually push it the same way you would normally push code. You can put it in the Git. Maybe you have some CICD tooling that gets that in S3. Then you could refer to that notebook object as the thing you want to execute later.

I would say most people actually at Netflix that are doing the notebook development are using our managed notebook servers. We have a suite of servers and you can like launch your own to say how much resources you want in your notebook server and you will actually boot up a container that has the notebook server up with all the things and solve that you might want to use. Actually mirrors the execution environment that scheduling that notebook would have.

But let's say either of these PaaS, you've got notebook either in S3, in Git, or in your notebook server. We actually mount these file systems and make them visible to any of the schedule jobs. For us, all you have to do to schedule one of these notebooks is copy the URL that you're looking at and then write a few lines of YAML and specify that URL is what you want to execute, and then we'll snapshot that notebook for you and then we'll run it based on when you give it to the scheduler that version of the notebook moving forward. Then you're able to do things, like independently tweak the resources you need or independently being be able to move how that notebook executes relative to other code.

[00:30:07] JM: There's an ecosystem of extensions to notebooks that you have taken advantage of in Netflix. Can you describe some of the extensions?

[00:30:17] MS: Yeah. There's quite a few, and I think this is actually still like an ongoing story. I would say there's a plethora of things you can add on to a notebook environment. We try to keep it generally minimal so that there's not too many things that are specific to one environment, but one set of tooling that we use quite a bit is around the Interact project. Interact is like an open source set of extensions or rather a rewrite of the frontend for some Jupyter Notebooks interfaces and it's React-based so that it can be componentized, and then these components can be shared and used in different places. A lot of other systems and tools that run notebooks also kind of pull some of these components and Interact designs.

We use that to give a better –

[00:31:03] JM: Sorry. As in React.js?

[00:31:05] MS: Yes, as in React.js. We wanted to kind of – If you look at the classic, some of the older notebook infrastructure for the frontend, it's built with older frontend tooling and it was really hard to upgrade that tooling in place. So a lot of teams and tools have sort of built new rendering frontends using modern technologies. We use those tools a lot of help give a better experience.

[00:31:29] JM: Okay. So one extension is Papermill. Papermill allows you to provide parameters to a notebook. Can you describe why would you want to parameterize a notebook and some use cases for Papermill?

[00:31:41] MS: Yeah. Actually, Papermill is the key to why we can schedule notebooks reliably. It's the key to kind of flip on its head some ideas around notebooks. So I describe exactly what Papermill does, one of the limitations with notebooks traditionally is that it's editing the document that represents what's it doing in place. So as I'm making code changes, as I'm running different cells, the notebook object is actually both periodically and when you explicitly save, saving back to the original document store. Which means if you make changes and you delete a cell and then you say, "Oh, shoot! I need that cell back." You better hope it's in one of your checkpoints, because it could be gone. So much like editing in an IDE without committing or something like that your code.

What Papermill actually does is – What it says is it says, "Okay, we're going to isolate the source of truth of what we're executing." So the input notebook from the outcome that came from running that notebook, and that combined with parameterization means that you can take a notebook and really make it a template that you can programmatically configure to execute against different targets, and then you can see exactly how it got configured and exactly how it ran each time without overwriting any of your sources of truth.

So the library is actually pretty small, but what the library does is it reads from a source, and we have many different sources you can read from, like S3 or Zeroblob or your local file system. It pulls those net file in, that JSON document, and then it finds the appropriate place based on tags or if it doesn't find any they'll put it at the top, a place to inject a new cell which has code. It represents all the parameters you pass. So you give it a bunch of JSON parameters. It's going to convert those JSON parameters into actual code that gets executed at runtime.

Then as each cell executes, it saves the document as it currently is out to the output path. Usually, we give like a unique output path based on the context of while you're running, plus a UUID, and we dump the notebook there. What happens is, is while it's executing and when it completes, you get to see both the visuals, the logs and the code that's set for exactly that run. You can always go back and take that notebook and reproduce what ran at that time by just – You can actually – We even put a button in there so you can just click clone to my local environment and it will clone to your notebook server and you can just play with it there.

[00:34:03] JM: Is it an important use case to be able to have a notebook that you write and execute it and then have the output passed as parameters to another notebook?

[00:34:14] MS: It is. It doesn't get used too often, but there's a few machine learning use cases where people will actually – So there's some tooling so much that I've been working on advancing, where you could actually save data in the notebook as you're executing. So you can do things like save the result. Like I did an experiment and I want to store what the confusion matrix was. So I can record that confusion matrix right in the notebook, and then if I pass that notebook to a particular place, I can actually read it later on from another source either in another notebook or another job and find out what the outcome of that notebook was by using this library to pull out the information that was saved.

That is one use case for beyond that where Papermill is used. Though I would say usually we end up actually persisting data that's important to tables in S3. For us, many times, if it's really important information and you want to pass it along, we have like other systems already in place that we plug into.

[00:35:05] JM: So instead of just passing – Like if you were doing – Let's say you had a notebook for a data cleaning operation where you – Or data enrichment, where you take a geo and you enrich it with the most recent location, or the nearest location, like according to Yelp or something the most closest restaurant so that you want to identify what restaurant somebody is sitting in while they're watching a Netflix movie and you're just getting the geo. So you're running a little notebook to calculate the nearest location.

You wouldn't want to just pass that geo to the next notebook and then do some more execution, like what other movie should we recommend to them given that they're sitting in McDonalds in North Dakota. You would want to save that data in S3 and then perhaps another notebook would just read from S3.

[00:35:56] MS: So because of the history of Netflix, we have a lot of mature tooling around managing tables in S3 or tables in other systems that source of truth from S3. So a lot of times we lean on that because we have a lot of good fundamentals there and it's well understood how it behaves, how you access it. All the tooling knows how to get to that. So we wanted to lean on not reinventing the whole world, but making notebooks a different way to enter into that world.

So, here, definitely we do a lot of data transposing, where you're going to enrich, augment, combine different tables and generate some either temporary or permanent outcome tables and then consume downstream, whether that'd be a tableau report or whether you have some custom visualization that reads it or some other notebook job, which actually reads that and does more further computation.

[00:36:41] JM: Can we give a few other examples to help illustrate this in people's minds when you're talking about data enrichment or creating dashboards for people. Maybe some real-world examples – I don't know, list of movies or statistics or something?

[00:36:57] MS: Yeah, actually. I was actually realizing I needed to – Earlier, you asked for more examples, and didn't give very many at the end of it.

[00:37:04] JM: That's okay. I gave you a double question. So I was the original sin.

[00:37:09] MS: Oh, good. Yeah. Here's a perfectly good example. So people have been using – And this is a good combination, notebooks and data. People have been using this new notebook template that they've generated, where they want to find out during a particular title when do people rewind. What parts of the title do people re-watch? What are the interesting parts of the show? What they did was they made a notebook, which runs a query that executes against our event table, which attracts different events that were taken on behalf of users in the system. That notebook, you can specify a title that you want to look at.

So one example we use was you go look at Frozen and you go find, "Hey, I found out what was running for Frozen." Then we actually gather all the click events, aggregate click events for that and we go ahead and find out, "Hey, how many users were watching it at any given minute of the show?" Then you can actually see – It looks like a linear graph going down, and then there's a few spots where if you back up, the plot jumps up for like two or five minutes. Those are the different places. If you haven't guesses, those are the places where the song start. People actually rewind – Somebody, probably everyone's kids. So the rewind over the songs that are really popular from that show. But it was something that got pulled out of the data by just looking at the title and seeing where people were watching more than other places. That's all done programmatically via a query at the top of this notebook followed by this template, which does a very nice visual display that shows the graph, the title pictures, and then generates links, which you could actually build as an embedded video if you want it in a notebook as a media type to be able to playback those sections that were popular.

[00:38:50] JM: Wow! This would be an example of a notebook template. Is that what you said?

[00:38:55] MS: Yeah. This is a template that people are sharing among like content teams that are trying to do analysis on shows.

[00:39:01] JM: The template being you can reuse this for other movies?

[00:39:06] MS: Yeah, absolutely, because it has an interface at the top where you select the movie – You like type in the movie you want. You can do the same thing by parameterizing it with the Papermill library we talked about before. So you could run the same notebook and just pass the parameter of movie title is Frozen, or whatever if you know some code of the name of the movie that you want to look at. Then it would run this scheduled in the background and then save the outcome some place. So you wouldn't have to run it iteratively. You could just go look at the end result.

But because it's a notebook document, you can actually still interact with that document when it's done. All the media types that you output, you can interact with on any frontend like you would normally with a traditional report.

[00:39:44] JM: You could also take that template and fork it and find out when people are fast-forwarding.

[00:39:50] MS: Yeah, exactly. You can make small tweaks, and that's actually a really common thing. We like want to promote the well-made notebooks so that users can take those, get ideas, iterate on them re-share them for new use cases.

[00:40:02] JM: Are these notebooks mostly used for this kind of exploratory data analysis or are they also used for prototyping machine learning models that are going to go into production and do things like give people recommendations?

[00:40:17] MS: I would say it depends on the team, but both. It's also used, as we mentioned before, it's actually now being used for all the batch work, like all your Spark and Pig jobs that used to run on the system via kind of older tooling is now also all running using the libraries that we've built via notebooks. But I would say that for analysis and machine learning purposes, it really depends on the team and the use case.

If you're doing something on your own and you're exploring an idea or a model, it would be very common for you to kind of try a notebook and then be running everyday on like small tweaks to that notebook where you're trying to figure out kind of the best outcome. If your team is already developing in a tool that's been here a little bit longer, you might still stay within your tool a bit more and make a model that runs in there.

Though we haven't seen a lot of interesting use cases where people are still using the tools that they're used to, but they're still wrapping it with a notebook so that they can separate out how they're operating. Then the move to production now is a different story that it used to be.

I would say, traditionally, at larger companies with big data systems, the story about once you get your model done is, "Okay, now the tedious part where we're going to rewrite everything you did in another way." Here, there're traditions we made. If your notebook is built in a way that is modular or parameterizable or easily, you can actually just tweak that notebook and then schedule it. We have a fair number of users doing that. We also have a fair number of users that are just staying within their ecosystem on the machine learning, because they want to make it more familiar to their neighbors. I think given the number of people that are using notebooks now, even that story is starting to change.

[00:41:53] JM: If I recall correctly, much of the machine learning tooling is written in Java, because it's like if you're using Spark, you're in Java, but there is an interface into Java or into Spark with Pi Spark. So are you saying that some of the machine learning users are staying within the Java world?

[00:42:13] MS: Actually, that's an interesting thing, because one of the big initiatives we've been working on is trying to make the Scala Spark experience better. The notebooks aren't isolated to only executing Python. This is maybe a misconception of how notebooks are set up. Notebooks are literally a protocol for executing code that are not language-specific. As long as you follow the protocol for sending messages back and forth, you can actually run with pretty much any language. A matter of fact, if you go to like the Jupyter page and look at all the supported kernels, there's maybe like 30 different language, different languages represented in those kernels. Kernel is a notebook representation of a program that can execute the code within a cell by a particular name.

For example, you might have a Python 2 kernel, or you might have a Scala Spark kernel or things like that. So I would say, yes, people are generally keeping it in their tool. The language boundary is becoming less and less of a thing for that purpose. So a lot of the Scala dev has actually been using Zeppelin Notebooks, which is a variant on notebooks separate from Jupyter that's just has a different take on how to approach the problem.

We've been trying to consolidate them on Jupyter mostly just so we have one notebook infrastructure and we're not supporting two. The other place where people may be haven't been using notebooks explicitly are with using – They are actually using Python, but they have like really good tooling, which has partial integration with notebooks and it's going to get more in the next year that allows them to define their flow of works that they're executing on and their model iteration gets kind of abstracted away from them. But even those tools end up using the same scheduler and using the same tools that schedule everyone else's work.

[SPONSOR MESSAGE]

[00:44:01] JM: Software engineers can build their applications faster with the use higher-level APIs and infrastructure tools. APIs for image recognition and natural language processing let you build AI-powered applications. Serverless functions let you quickly spin up cost-efficient, short-lived infrastructure. Container platforms let you scale your long-lived infrastructure for low cost, but how do you get started with all of these amazing tools? What are the design patterns for building these new types of application backends?

IBM Developer is a hub for open source code, design patterns, articles and tutorials about how to build modern applications. IBM developer is a community of developers learning how to build entire applications with AI, containers, blockchains, serverless functions and anything else you might want to learn about.

Go to softwareengineeringdaily.com/ibm and join the IBM Developer community. Get ideas for how to solve a problem at your job. Get help with side projects, or think about how new technology can be used to build a business. A softwareengineeringdaily.com/ibm, you will find the resources and community who can help you level up and build whatever you imagine.

Thanks to IBM Developer for being a sponsor of Software Engineering Daily and for putting together a useful set of resources about building new technology. I'm always a fan of people who build new technology, who build new applications, and this is a great resource for finding some design patterns and some new ways to build and leverage these technologies, like serverless, and containers, and artificial intelligence APIs.

So thanks again to IBM Developer.

[INTERVIEW CONTINUED]

[00:46:00] JM: When you're running some of these data science jobs in a notebook, they might be very resource-hungry jobs, like a Spark job, you might be looking over – I don't know, terabytes of data and doing analysis on that, and in order to do that, you need to have a way of deploying your notebook to run across a Spark cluster, I believe. Am I getting at why you decided to build all these scheduling infrastructure around notebooks?

[00:46:32] MS: Yes, because what ends up actually happening is your notebook can run maybe two different modes. You might either rerunning that notebook, where you're doing all the computation locally. Like I have something that's collecting all the data, running it within the notebook on the server that's running the notebook. That might be something like where I would put data in, say, a Pandas data frame and do some analysis with some tools on it and then come to a result. This one execution pattern was just common.

Then the other pattern is actually where your notebook is really an integration tool. It's defining what other systems and ecosystem tools you need to connect to to actually do the real work and then orchestrate it all from that notebook.

So, for example, we have templates that let you spun up your Spark job, or your Presto job, or Pig job or any of these other types of systems where you're really defining the inputs, like I'm defining the code that I want to run, and that could even be in different languages, SQL, Scala, Python, and then I'm defining like all the configuration.

Like it needs to run in this cluster because I know it has enough resources. It needs to set the driver and executor memory. Everything you would think from an ETL system, which can kind of give same defaults for in this notebook environment using the libraries that we already have for other tools.

Then when they actually execute these notebooks, many times these notebooks are really just running jobs on the ecosystem we've already built for doing other work. So if you're running and you want to federate out your Spark job, you're going to use a same job federation tooling under the hood that all the other jobs that are federating work on the platform use. It's just the calls to the libraries, which makes that job run are now being represented in a notebook, and then we can collect results and visualize them a little bit better automatically for you.

[00:48:15] JM: Give an example of a workflow that you would need to schedule for a notebook. Something that would consume lots of data or lots of resources, and take me through the execution path of this distributed scheduler infrastructure.

[00:48:31] MS: Yeah. Maybe I'll give you one example, where it's actually you're re-templatizing the same notebook many different times and you're distributing the work. So say if for example you need to do some analysis on what shows are being watched in each different region. Let's say every country code or even down to maybe say state or county or some other delineation of space. So you have a notebook which says, "Hey, given a particular region, particular set of days, I can tell what's the most popular show in that region broken down by a few different factors," maybe by say the length of the subscriber, how the long the subscriber has been on there. Maybe what else they watch. Things like that.

That might be a pretty heavy job. So if you think about the number of events in the system, we're getting trillions of events a day. That means you have very large tables behind this. So it's not ideal to be pulling those tables down all the time.

In this case, you might make that book, which, "Hey, I'm going to isolate this so that the notebook will run for a very specific region. I'm going to run it just for let's say the U.K. and I want to have it for the last 28 days all the different views and these different breakdowns." Then I run that, and it's great. I've got an answer and so the content producers there can use that. But then let's say when you want to do this for all the regions, because actually Netflix tends to do changes or updates or new releases globally all at once. So you want to really get a global view of things, but that's too much data to run in one place and it's a lot of information to come back.

So one thing we actually do here is say you've got this working for one region and you want to run it for all regions. Well, what you can do is we actually support the ability to fan-out in your orchestration where you would say, "For each region in some long list of regions that I want to run, run this notebook passed the region that you're currently iterating on and any other configuration you want to pass." Let's say we're going to focus on views in the last 28 days. We're going to say, "Hey, the time range is 28 days. Particular region is region K, and now execute this notebook."

That might fan-out to 100, 200 different notebooks that are running the same exact notebook with these just couple of minor configuration changes, and you can let those all run in parallel and then collect the results at the end. Then at the end, the notebook there would say, "Hey, I'm going to go read the outcome results from the tables that each of these notebooks generated and I'm going to build an aggregate view that lets the user select through these."

At this point, maybe I would do that in a notebook. Maybe at this point I would actually funnel that data to another report that already exists someplace else, but in that sense you've created the one template notebook that you've been able to develop locally on gigabytes of RAM, and now you're distributing the data across terabytes. Even within each of those particular notebooks that are running, they actually can be set up to run – This computation is really being run as a Spark job or as a Presto job so that that computation is being funneled out to the compute cluster.

This is sort of like an ecosystem view of like this is actually a real workflow I'm actually describing that someone builds like the moment we had notebooks available, we have this fan-out option, and then they started running this so that every week they could give this report up to their team that said, "Hey, what's the value add over the last 28 days of different shows?"

[00:51:47] JM: Right. I can imagine just this workflow where I'm a developer, or I'm a data analyst and I am just doing this on my local machine and I'm building some kind of report and building sort of the N=1 case for this report, and then in order to run the case for all the data, then I would just execute it across the data infrastructure which would be a lot more resource intensive. In order to do this, you use a scheduler called Meson. Can you talk about Meson and why you built it?

[00:52:25] MS: Yeah. When we actually looked at the scheduling problem, there was a lot of different options and a lot of people were using different schedulers that have different pros and cons. We also looked pretty heavily at using Airflow. We actually had support for Airflow for quite a while, and the servers were up for all of last year that we were kind of playing with, but we ended up going with Meson, and Meson was an internal scheduler tool that was really designed around machine learning infrastructure needs.

So it has a lot of features around this like fan-out or this conditional execution, and the integration with the platform for that tool is better than the other tools we had. We actually ended up choosing that one mostly because the open source options that we saw didn't solve all the needs we wanted anyway. This one already have platform integration and a team that have been developing on it.

So we ended up leaning into that tool itself to solve our workflow execution, and we actually build a layer on top, which was more oriented around how we wanted to think about this data execution with notebooks and with very trivial config options in order to specify what you want to run. It's a workflow executor. If you think about it, it's just executing a DAG of work where you describe nodes, and those nodes could be notebooks or they could be just arbitrary containers to execute.

[00:53:42] JM: So I guess I didn't quite understand why did you go with Meson over Airflow?

[00:53:49] MS: So in that case, the end point that we ended up choosing to is, one, we didn't have a ton of Airflow users already, and this tool already had a user base. The integration with the rest of our platform and its tools and its identity and all of those types of things was already partially or completely present on Meson. We ended up choosing it really, because the tool was the most likely to succeed in the next year for the project rather than because the tool had completely superior feature sets.

The reality, like we could have gone with Airflow and been perfectly fine just as an investment there, and the investment would have been really for Netflix specific things that wouldn't been helpful to the open source community either way. So we chose one.

[00:54:29] JM: Can you talk more about the importance of these kind of DAG tools, the tools like Airflow or Meson and why they're so important to a data infrastructure?

[00:54:41] MS: Yeah. So it's extremely important to be able to specify that work, which is if you're not familiar with graph theory, that's a directed acyclical graph, which really means it's a tree that doesn't circle on itself. The reason why DAG has become important, because often times you have these pipelines or workflows where you need to basically do some initial aggregation or initial computation or data collection and then you're going to have different downstream work that relies on that work being completed.

For example, if you need to send an email report on the status of some state of data, you usually have to run a query first, which generates that data. You might have to join it with other data. You might have to put it in a particular system and then you might have to query from that system. There you might have a complex set of four or five different units of work that you actually have to execute and execute in a certain way with certain dependencies both on jobs and on data. That's really where specifying a workflow in a DAG becomes very useful, because you can orient and visualize how the work is going to execute ahead of time and afterwards in a way that really facilitates this kind of dependency-driven data pipelines.

[00:55:55] JM: I've talked to a number of different companies about data platform, whatever that means at different companies. So at DoorDash, it means one thing. At Uber it means something else, but it's something similar. You've got data that's coming in as online transactional data and then often times that data is ETL'd into a place where it can be analyzed at large scale and there's lots of difficulties around that data platform. It seems like notebooks solves some aspects of the data platform. Can you talk about the other predominant issues that you're seeing in the Netflix data platform or what you've seen at other companies?

[00:56:40] MS: Yeah. It's an interesting problem. A lot of companies have a lot of the same pains with data platforms, and it's really a problem that distributed systems are complex and hard to do. So the problem is, is that distributed systems are hard and complex to do. I would say some of the key points where we still see a lot of friction is even just – What we're working on a lot this year in continue in the next year is like visibility and discoverability of what data is important and of understanding when something goes wrong or when something is going to go wrong, how do catch that and remediate it efficiently and easily.

That's definitely been one of the larger pain points. I would say for less mature platforms and however you describe platforms. As you said it's a little ambiguous. I would say that there's more emphasis on getting tooling that supports users to be able to do what they need to do. We still have that problem. I think it's an ongoing problem for everyone. But definitely when you get to the phase pass where, "Okay, the basics are possible. Everything is kind of there." The next you have is, "Well, this works and I can keep track of it at this scale," but the next 10X scale or the next 100X scale, it gets harder and harder to get visibility into where are the problems, why is this job not running? Why is data not ready? You start getting a lot of these sort of critical PaaS through your data pipelines that are really important that they execute on time or on a particular cadence.

So those areas were getting like awareness of where work needs to be done to improve or to monitor or to track what's happening is a really critical thing that even the best companies still are working furiously on solving.

[00:58:16] JM: As we begin to wrap up, I just want to leave people with kind of the important takeaways for what notebooks have done for Netflix. What I have understood from this conversation is that they really help with the process of running jobs that, for example, create detailed reports and maybe they're highly interactive reports. There are reports that have videos embedded in them that show, "Here is where people rewind for the movie that you just released on Netflix yesterday." Maybe you want to send out that report to everybody that was in charge of a content team that have new release on Netflix over the last week. It's useful for that kind of thing.

It's also useful for people that are maybe developing locally on a data enrichment tool. Maybe they're enriching – They're just enriching like a couple of movies on their local machine with, for example, the geo that it is most watched in and they want to add that to a field in an S3 bucket, but except they want to do that for all movies. So they test it on their local machine and then deploy it to large scale. Are there any other use cases, any other examples that people should take away for why these things have been so useful in Netflix?

[00:59:28] MS: Yeah. Actually, there's one more that's been extremely useful, and this is one of the ones we refer to as a strategic bet as you talked about earlier, and that's around the reproducibility and visibility on what's executing in the system. It gives a huge benefit there. Let me describe that for a second. So if you have a notebook which ran and somebody wrote it, and that person may not even be at the company anymore or it's on a team that the person is on vacation or no one could be reached and it goes wrong and there's five things downstream that need to use that notebook.

Traditionally, when a pipeline goes wrong and you have to start debugging it, you have to go find out where their code is, go read their code, try and parse what that's doing, backup to whatever libraries they use and you might spend a long time trying to explore what actually executed. How did it run? Which log line corresponds to what line of code in this execution? So it can really slow down and make it very tedious to be able to have other people support work that's been written by someone else in the company.

One of the really big benefits here with the notebook is we have a very consistent interface for what executed. You get this artifact, which is both visually oriented and has the code that ran right next to it. So I can see for a particular log line exactly where it came from. I can see exactly any kind of debug information we put is like associated with the lines of code that was executing and it's a familiar template for everyone. You don't have to figure out what language this script is written in. What language is that thing written in, or even where it is. You get a familiarity of evaluation on what actually ran, and it's like the first link we go to debugging anything. It's like, "Okay, what's in the notebook?" You click it and you go get to visualize what's there.

Sometimes if it's confusing, you actually pull that notebook and run parts of that code to understand what it was doing and why it failed, and that's given us a huge boost to be able to support different users without having as much training or specialization.

[01:01:18] JM: You've built a lot of infrastructure for auditing and debugging notebooks when they run and something goes amiss.

[01:01:24] MS: Even there, there's not much tooling needed. Mostly it's just – We point to the notebook and a little bit of tooling so that you can launch that notebook in your own ecosystem.

[01:01:32] JM: What advice would you give to somebody who's listening to this who has data problems at their company and they want to start using notebooks, or they want to start putting notebooks at the center of the development process at their company?

[01:01:44] MS: Yeah. I would say the first thing to do is really actually don't force all your users to immediately have to develop in notebooks. Make notebooks an optional path, where maybe you start doing all of your ETL templates with notebooks and you very closely control a few special notebooks that you share with people to use and you may be make it so that the user doesn't have to know they're using a notebook to get started. That actually was really successful for Netflix.

Everyone is using notebooks all over the place, and then everyone else is also using notebooks and they don't even necessarily know until they start debugging or they want to like slightly change how one of these templates works.

That's been a good like natural intro to it, and I think having a few very controlled paths where you're having an integration problem that a notebook would help describe well and that you could share is a good place to start. I've launched a tooling, I need to talk to three systems and may have some outcome, and it's going to be a page of code. That's a great place to find that in a notebook and to lean on the libraries you've already developed.

[01:02:40] JM: Matthew Seal, thank you for coming on Software Engineering Daily. It's been really fun talking to you.

[01:02:43] MS: Yeah, definitely. Thank you for having me.

[END OF INTERVIEW]

[01:02:49] JM: How do you know what it's like to use your product? You're the creator of your product. So it's very hard to put yourself in the shoes of the average user. You can talk to your users. You can also mine and analyze data, but really understanding that experience is hard. Trying to put yourself in the shoes of your user is hard.

FullStory allows you to record and reproduce real user experiences on your site. You can finally know your user's experience by seeing what they see on their side of the screen. FullStory is instant replay for your website. It's the power to support customers without the back and forth, to troubleshoot bugs in your software without guessing. It allows you drive product engagement by seeing literally what works and what doesn't for actual users on your site.

FullStory is offering a free one month trial at fullstory.com/sedaily for Software Engineering Daily listeners. This free trial doubles the regular 14-day trial available from fullstory.com. Go to fullstory.com/sedaily to get this one month trial and it allows you to test the search and session replay rom FullStory. You can also try out FullStory's mini integrations with Gira, Bugsnag, Trello, Intercom. It's a fully integrated system. FullStory's value will become clear the second that you find a user who failed to convert because of some obscure bug. You'll be able to see precisely what errors occurred as well as the stack traces, the browser configurations, the geo, the IP, other useful details that are necessary not only to fix the bug, but to scope how many other people were impacted by that bug.

Get to know your users with FullStory. Go to fullstory.com/sedaily to activate your free one month trial. Thank you to FullStory.

[END]