EPISODE 447

[INTRODUCTION]

[0:00:00.6] JM: It's 9 p.m. at night and you are hungry. You order a pizza from Domino's. You live on a street that's dark, so you have installed a smart light bulb in front of your mailbox that lights up the address. When the pizza at Domino's is ready, you want the light bulb on your mailbox to light up so that the delivery person could read your address when they arrive in front of your house with the pizza.

The internet should make it possible to have this kind of event-driven connected world. Anything that is connected to the internet should be able to send signals to anything else on the internet so that our lives gradually become more automated. This is what IFTTT does, if this than that, otherwise known as IFTTT.

Users of IFTTT can easily create applets to wire different services together. You can use IFTTT to trigger an email whenever three of your friends or retweet something on Twitter. You could use IFTTT to flash the lights in your house when Bitcoin hits new market highs. You can use IFTTT to send yourself a pizza whenever Bitcoin crashes. IFTTT makes it easy to connect different services together, and a lot of work goes into the infrastructure that enables these billions of events to process correctly.

Nicky Leach from IFTTT's engineering team joins the show today to describe how IFTTT allows for integrations between services that were not built to integrate, and he talks about the scheduling, the data engineering and the monitoring of the company's software stack.

[SPONSOR MESSAGE]

[0:01:38.1] JM: When your application is failing on a user's device, how do you find out about that failure? Raygun let you see every problem in your software and how to fix it. Raygun brings together crash reporting, real user monitoring, user tracking and deployment tracking. See every error and crash affecting your users right now. Monitor your deployments to make sure that a

release is not impacting users in new unexpected ways, and track your users through your application to identify the bad experiences that they are having.

Go to softwareengineeringdaily.com/raygun and get a free 14-day trial to try out Raygun and find the errors that are occurring in your applications today. Raygun is used by Microsoft, Slack and Unity to monitor their customer-facing software. Go to softwareengineeringdaily.com/raygun and try it out for yourself.

[INTERVIEW]

[0:02:44.2] JM: Nicky Leach is an engineer with IFTTT, if this than that, otherwise known as IFTTT. Nicky, welcome to Software Engineering Daily.

[0:02:53.2] NL: Thanks for having me, Jeff. I'm excited to be here.

[0:02:55.2] JM: Today we're going to talk about IFTTT and the backend infrastructure that powers it. Explain what IFTTT does for people who don't know.

[0:03:05.6] NL: Sure. There are sort of two sides to IFTTT. As a consumer, IFTTT allows you to connect all the different things in your life and make them work together, an those things might be like connecting your Twitter account to your Facebook account, or your connected car to your lights at home. It provides easy functionality for users to build those connections themselves.

The other side is that we have a platform for companies to integrate with IFTTT provide those integrations for their end users. It's a really easy way to have one sort of integration work with, today, over 550 other services on the platform.

[0:03:41.2] JM: There are these two abstractions; service and applet, that people can build on IFTTT. What are these two terms mean?

[0:03:50.0] NL: Definitely. As a service is sort of the term that we used to describe either something like an interest service, again, like twitter, or maybe an internet connected product,

like your light bulb or your car or Domino's Pizza. An applet is sort of a unit of functionality that integrates with one or more different services. You might have an applet that will post your — This is sort of like the canonical example, posting your Instagram photos to Twitter directly as opposed to having just like the Instagram link, which doesn't show up in a lot of Twitter clients. Users turn on individual applets or they can create their own to create those sort of mappings between different services.

[0:04:32.4] JM: Okay. Give a few more examples of services and applet so people can understand how these different components fit together.

[0:04:42.5] NL: Right. Services are sort of like any sort of brand of property that you might have an association with. Again, sort of like on the social side; Twitter, Facebook, Instagram, LinkedIn. In terms of internet connected devices, like your Hue light bulbs, your LIFX light bulbs. We have a couple sort of like more novel service integrations like, again, Domino's pizza. It's either you have this pizza tracker service where you can see the status of your pizzas. It goes through the preparation and delivery pipeline and then do different things. You might have an applet that says, "When my pizza is out for delivery, turn on my porch lights."

[0:05:21.2] JM: Right. What are some of the apps or applets or services that you use in your everyday life? How does IFTTT fit into your daily life?

[0:05:30.4] NL: I'm a total Slack addict, so a lot of my applets are around like curating content in Slack. We use it around the office a lot for things like standup reminders and things like that. I also have recently started doing a little bit more of like life cataloging, and a really interesting way to do that is to take different events that you find, I guess like input sources that you find meaningful and having the output be a calendar.

Things like your check-ins on Foursquare or Swarm and having the output be a calendar. Things like your check-ins on Foursquare or Swarm, we have this awesome location service where you can set up triggers for when you enter or exit given areas, like when I get to work. Having those all end up on a calendar, it really gives you a good sense of sort of like what's going on in your life retrospectively.

[0:06:15.2] JM: How is the rise of the voice interfaces affected the IFTTT business?

[0:06:21.9] NL: It certainly brought us a number of users. Amazon Alexa is certainly one of the more popular services on the platform, and I think that it's a really natural pairing. Before there were any or even many skills in the Alexa ecosystem, there are tons of integrations that you could use with IFTTT. I know that when Amazon launched in Europe, IFTTT users were just like clamoring to get support in the products.

I think that there is a natural alignment between sort of, again, that like one to many integration that users are able to really apply creativity and build interesting integrations.

[0:07:00.4] JM: Okay. Let's start to talk about what's happening on the backend. When I stand up a new service, and that service integrates, how does it connect to IFTTT's end? Are you standing up an API server for that service or what exactly is going on?

[0:07:19.6] NL: Definitely. I think we can sort of answer that question starting historically. Seven years ago when IFTTT was started, all those initial integrations were just built directly into our model. I think application is just like API integrations using different client libraries and things like that. It didn't take long for us to realize that that just like really doesn't scale in terms of our ability to integrate with as many services as possible, also to have a deep understanding of the best ways for those services to work together. Really, the people who built those services have f the best understanding of like the valuable data, what interactions are powerful on their platforms.

About three years ago now we started building the IFTTT platform. The API side of that is what we call the IFTTT protocol, and it's really just a standard set of HTTP endpoints that you have to implement to integrate with IFTTT, and there is like a schema that basically defines what a trigger should look like, what an action should look like and a bunch of tooling around making sure that those endpoints actually sort of fill that specification from sort of a soup to nuts perspective. There are two different ways that partners tend to integrate with the platform. With their IFTTT integration being sort of directly inside of their application, sort of like making database calls directly or whatever it might be, or the IFTTT integration being sort of like a shim service in between their either public or internal HTTP APIs and the IFTTT protocol.

[0:08:53.7] JM: Can you walk me through what happens when someone makes a call to a recipe? I guess we actually have not explored the terminology recipe. we talked about applet. We talked about service. Describe what a recipe is. Maybe you could refresh people on what a service and applet is while you're doing that. Eventually, we'll get to a more detailed description at the backend architecture. I just want to give people a really good understanding of what the frontend looks like.

[0:09:25.2] NL: Yeah. I think that thinking about sort of the important IFTTT vernacular, services are brands, they're like logos, they are things that you have a relationship with. Applets are a specific functionality. Again, sort of if; this, then, that statements. That's where the name comes from. Applets are sort of a rebrand of a previous concept called recipes, and it was really sort of — There are some functional differences, but for the most part it was sort of a marketing decision.

The idea being like we want — Recipe implies work, like recipes make baking a cake a lot easier than trying to figure that out on your own, but you still have to go through the steps of baking the cake. Applet is sort of the idea that we want to communicate is that there are little apps that you turn on and you don't have to do a lot to get that.

Then inside of an applet, there's going to be more one triggers and actions, and a trigger is a new piece of data coming into the system. That might be you got to a specific location, you posted a tweet. A new entry was created in a database somewhere. Then action is creating content, doing something. That might be sending a push notification or turning on a light bulb.

[0:10:44.8] JM: The service where I upload a picture to Instagram and instead of just copy pasting the link to Instagram into Twitter, if you instead wanted to have the picture itself render in the Tweet. This is a really common use of IFTTT as you already mentioned. This specific use case of Instagram linking up to Twitter, this would be a recipe — Well, sorry. An applet, formerly called a recipe.

In this example, Twitter and Instagram would both be services and you're linking together these services with and applet. Have I described that correctly?

[0:11:29.0] NL: That's 100% correct. You did it better than I could have.

[0:11:32.7] JM: Okay. In calling that recipe, there are some pieces of data that are being passed to the recipe. Where are Twitter and Instagram plugging in and listening for that event and what pieces of data do they need from that event?

[0:11:54.4] NL: Definitely. The most basic level, this is maybe getting a little bit into like how IFTTT works behind the scenes is. IFTTT is calling APIs to check for new data constantly millions of times a minute effectively. In this specific case, like Twitter, or Instagram to Twitter, IFTTT is pulling Instagram's API to say, "Are there new photos? Did Jeff post a new photo?" When we detect a change, we take that data, package it up into sort of an understandable set of pieces of data we call ingredients that the user can configure to say, "Semantically, this photo URL, I want that to be the thing that feeds the image that I post to Twitter." So then we do that mapping, and then on the other side of the applet execution, we're sending that data to Twitter using sort of effectively going through the Twitter's public API.

[0:12:54.6] JM: How much volume is coming through IFTTT. How many calls to different applets are you getting?

[0:13:04.1] NL: We are on the volume of about a billion API calls a month, and most of that — This is sort of like the dirty secret of IFTTT, I guess. Most of that is us polling to see if anything has changed, and in almost every case nothing has changed. The total volume of API calls that result in some desired outcome is much lower, but we keep up that volume in order for things to happen as quickly as possible in places where there isn't a real-time integration.

[0:13:32.6] JM: It's unfortunate that you have to do the polling instead of there being some web hook or web task on the providers on the services and where they could have their web hook be scanning the event stream for things that they would want to publish to IFTTT. It's kind of unfortunate. It's a lot of wasted effort because of that polling.

[0:14:02.3] NL: We do have a real-time interface in the protocol, and what that allows our partners to do is let us know that new data is available for us to fetch. Due to a number of

circumstances, often just like the functionality not being there in the origin API, the default behavior is to just pull on regular intervals for that data, but for something like Amazon Alexa or the latency in that exchange is so important. They definitely use a real-time API to make sure that that data enters the IFTTT system as fast as possible.

[0:14:37.7] JM: Okay. Whether the the service is sending you the data or if you're polling for it, eventually you get notified of some new event, and that happens on a very regular basis. If I understand the architecture correctly, the first place that the event is going to get registered is in Kafka. Can you describe where the API call is going to begin in your architecture?

[0:15:11.7] NL: The way that it works today, we do these checks on — I guess it's easier to talk just about the polling interface first and then we can sort of layer on top real-time. We perform checks on a per applet basis. If you have your applet that says, "Anytime I post a photo in Instagram, post that to Twitter."

We in-queue that applet on some schedule that we determine, and we can put a pin on that and come back to it. Then once that applet is checked, we make the API call to Instagram, check to see if data is available. If it is in line, that goes to the next phase where we post that data to twitter. There isn't today a disconnect between like the ingest and like action functionality. That ends up being just like one synchronous for — I don't know, continuous process that happens on an applet-by-applet basis.

[0:16:13.3] JM: What role does Kafka play in this architecture?

[0:16:17.9] NL: We currently don't use Karka. We do use Kinesis. We use it mostly as a tool for — Like our data and analytics pipelines as well as some of our other systems where we just need the sort of like a queue of data that is super durable. One of those cases is our activity feed system, where every outcome from an applet being run is output to an activity feed that a user can look at to see, "Did things work? If not, what was the failure?" Other interesting things around this data account goes through that same system?

[SPONSOR MESSAGE]

[0:16:58.4] JM: Heptio was founded by two creators of the Kubernetes project to support and advance the open Kubernetes ecosystem. Helptio unleashes the technology-driven enterprise with products that help customers realize the full potential of Kubernetes and transform IT into a business accelerator.

Heptio's products improve the overall experience and reduce the cost and complexity of running Kubernetes and related technologies in production environments. They build products, they build open source tools and they provide training and services and support that bring people closer to upstream Kubernetes. You could find out how Heptio can make Kubernetes easier at heptio.com/sedaily.

Joe Beda and McLuckie, who are the founders of Heptio have both been on Software Engineering Daily and they were fantastic when they came on. They're really fluent in what is going on in the Kubernetes ecosystem because they helped build it. To find out more about the company that they're building with Heptio and to find training and resources and products built for Kubernetes, go to heptio.com/sedaily. Thanks to Heptio for being a sponsor of Software Engineering Daily. I'm really proud to have the company onboard.

[INTERVIEW CONTINUED]

[0:18:27.3] JM: Okay. Maybe I misread something or I read an outdated blog post. I thought there were some main buffering system that Kafka was playing a role, but I guess you don't have really a main buffering system, like you don't have a central event log for all of the applets that are getting triggered.

[0:18:51.3] NL: Not is sort of like the critical path of applet execution. Sort of if you were to look at the output of our like our applet runtime time that ends up being used for analytics, you would see that data. Today, it's structured around this sort of like one-to-one mapping between — Or one to many mapping, I guess, between like an individual trigger track and actions being run.

[0:19:14.7] JM: Another piece of the architecture I thought I read about at least was GraphQL. Are you using GraphQL? Are you using GraphQL?

[0:19:20.7] NL: Definitely. We use GrapQL really heavily to power our client experiences. The entire data model is exposed via GraphQL and that is used by our mobile apps and our web frontend to build really rich interfaces.

[0:19:37.5] JM: Tell me more about that.

[0:19:39.2] NL: It's certainly a newer piece of our architecture. About a year ago, maybe a little bit more, we went through sort of like a UI overhaul. This is when we introduce the concept of applets. Part of that was trying to bring a little more consistency between the web and mobile experiences, and to do so we wanted to have like a more consistent interface between them. So many web applications, we were calling directly into the database to render pages and our mobile apps have to use REST APIs, and there was a lot that worked really well about that, but it made it difficult to always keep those experiences in synced. We made a bet on GraphQL that we could use that as sort of like the lingua franca between our different frontends, and that has worked out really, really well for us and allowed us to decouple sort of the data model itself or like where that data is persisted, things like that, from our ability to effectively use it on our web properties, our mobile properties. We've even used it internally for some of our monitoring systems and things like that.

[0:20:45.8] JM: People set up recipes on the mobile apps, and GraphQL is the linkage between the mobile apps and the backend. Is that right?

[0:20:59.2] NL: Yeah. I think that like — For folks who aren't familiar with GraphQL, you can just sort of imagine it as your typical REST API. The big difference being that the client is able to make very specific queries about the data that it's looking for and the relationships between those pieces of data.

On the mobile, you might be requesting information about the applet, and in the mobile UI you want to — In the same view that you have like the title in the description of the applet, you also want to show the number times this applet has been run for a given user. On a website, we might not want to do that. I'm totally making all of these up, but sort of as like an example. If the call to get the number of times an applet has been run is expensive, we don't want to be doing that on the web if we don't need to, if it's not like being shown to the user providing any value.

Using GraphQL, we're able to really precisely target the exact amount of data that we want without being wasteful, without downloading extra bytes over the wire, without making database calls or external calls that might not be important for a given transaction.

[0:22:12.7] JM: Okay. I want to breakdown in more detail the process of setting up an applet and servicing an applet request. I think we should figure out a different example, because I think some people — The Twitter-Instagram thing is a very familiar recipe for me, or an applet for me, but I think there are probably some people who are listening who do not use either of these services, that they have no idea what we're talking about. Maybe the Domino's plus Phillips Hue light bulb example would be better, where let's say I order a pizza on Domino's and let's say I live in a giant mansion and I've got all these lights, these Phillips Hue connected light bulbs that are lining the walkway to my door and I want to have an applet that makes it so that whenever my pizza is ready from Domino's, it's going to turn on all the lights in my walkway, because the Domino's is right around the corner from my palatial house and I want the lights to be turned on whenever the Domino's delivery guy comes by. Yeah, I want this applet to be able to connect Domino's to my Phillips Hue connected light bulbs, and IFTTT is going to be the service that brokers that. First of all, is that a reasonable example for us to explore?

[0:23:37.1] NL: It's a really exciting picture that you painted. I just want to callout that for people who are listening, there's a lot of value in these integrations, even if you don't have this amazing palatial mansion. People who just have like a light on their front porch might find it valuable too. Yeah, for sure. I'm totally onboard with this one.

[0:23:55.6] JM: Okay, great. Then, first of all, when I set up that integration, what is happening on IFTTT's servers?

[0:24:04.9] NL: Sure. I think the this is such a common use case. Domino's thought of this when they were building their service. They want people to be able to be ready to get their pizza. They want their delivery people to be able to see like that there are steps leading up to your door, whatever it might be.

Domino's has created — Has published an applet that says, "Turn on my porch lights when my pizza is out for delivery." As a user, if you're on ifttt.com or using our mobile apps and looking at

things that you can either do with Domino's or with your Philips Hue bulbs, you might see this applet, and it should be as simple as turning on the applet. Literally, it's a switch that you slide and it's on.

The configuration depends a little bit on the amount of sort of customizability. The author of that applet intends for you, the user, to have. In this specific case, there is no need for you to customize anything about the Domino's configuration, like a pizza that's out for delivery is a pizza that's out for delivery. You might want to have the option to specify which light bulbs you want to have turned on.

As part of turning on this applet and configuring it, the user is presented with a list of their light bulbs, and you would choose the porch light as supposed to your bathroom light, or you could choose both, I guess, depending on your use case.

Once you have configured the applet to your taste and make sure that it makes sense to you, the next time that a pizza is out for delivery, that functionality will automatically — That experience will happen for you.

[0:25:44.0] JM: On IFTTT's servers, there is just a polling process that's listening for the Domino's event, or Domino's is notifying IFTTT in real-time.

[0:25:55.4] NL: In this case I just know this, Domino's does use our real-time system. Within seconds of the order being out for delivery, early step being represented in the Domino's system, your porch light should turn on.

[0:26:08.2] JM: That's because Domino's has some user ID that they associate with my order, and so that they can publish that real-time event and IFTTT can listen for it.

[0:26:18.7] NL: Exactly.

[0:26:20.3] JM: Okay. Once that hits IFTTS's servers, I guess you can process the event and propagate it to the Phillips Hue light bulbs, and voilà, and I guess your lights are turned on. Maybe you could talk a little bit about the software infrastructure that is powering the connection

of those APIs, kind of your server infrastructure and how it scales up and down and whatnot, and then we can talk about some of the data infrastructure that allows for analytics on top of that, but I would like to understand just the run time infrastructure first.

[0:26:56.1] NL: Definitely. Our entire structure today is on AWS. We've been there for three years and are generally I think as happy as one can be with the cloud. We're a little bit unique, and that for all intents and purposes, our entire workload runs on AWS service called EC2 Spot. What that allows you to do this sort of like bid on excess capacity, excess EC2 capacity and acquire servers for sort of a limited amount of time at a really reduced rate.

To support having — I guess the risk there is that your servers may come and go relatively quickly. To support our infrastructure being able to handle like servers coming in and out of commission, we use Apache Mesos to manage how our individual server tasks, like application instances are deployed and run.

We have a large fleet of these Mesos nodes that are just effectively sitting there waiting for work to be given to them, and they'll run the work that's been distributed by Mesos until the task goes away. This is across all of our application infrastructure both our backend and our frontend experiences. I think one of the sort of misconceptions around Spot is that you're like never going to have an instance for more than an hour. In our experience we have Spot instances that stay online for weeks at a time and we're paying 20%, the sort of like on-demand price for those instances. It makes a lot of sense for us.

Then on top of that, like our sort of like application mesh. On top of Mesos we use — I guess, Marathon. I don't know who owns it anymore. It's Mesosphere. It's like an application scheduler. If you're familiar with something like Heroku, it's very similar. There is like a list of applications that are running. You can configure them with the environment variables. You can scale them up and down. A little bit less like fully featured than something like a Heroku, but it's pretty familiar analogy for a lot of folks.

We have dozens of applications that are running, and then across those applications, thousands and thousands of individual application instances that are being managed by Marathon and Mesos. Then for things like period background work, we use a another framework on top of

Mesos called Chronos, Chronos Kron. If you have something that needs to run every hour, Chronos is a really good tool for that.

[0:29:28.8] JM: Yup.

[0:29:29.3] NL: The individual — Like our deployment artifacts is Docker image, or Docker container, I guess. Again, all of our applications across the infrastructure run in Docker containers. We have a CICD pipeline built on top of Jenkins that will run any test for a given application as poll requests are made and then poll that emerged into master, and then our developers have command-line access to tools that allow them to tag an individual commit for a deploy, and that goes to sort of a similar process of making sure that that commit has been tested, that once that build is green from the testing perspective, we publish a Docker image and then give that configuration information to Marathon and Chronos and then those tasks are deployed.

[0:30:17.9] JM: When I think about how I would build the IFTTT infrastructure today if we were building it from scratch. The first thing that would come to mind would be serverless; AWS lambda or Google cloud functions, because the call to a recipe seems like a stateless one time spin it up and then spin it down type of event where you can get the functionality you need even from the extremely low cost infrastructure of an AWS lambda. Maybe you could tell me if that's true and whether or not you see that as a promising application of AWS lambda for IFTTT maybe in the future.

[0:31:00.0] NL: We do use Lambda internally for a couple of different things. Most notably, we now allow you to — As a power user effectively, to write a little bit of JavaScript code that sits in between a trigger and an action and an applet. You can do things like — In our Domino's to [inaudible 0:31:20.7] example. Check to see if the time of day in the user's time zone is like — I don't know, in the middle of the day. If it is, we don't need to turn on the light, something like that, right?

We use Amazon Lambda to execute that JavaScript code that our users have provided to us. It's just like a really simple way to sandbox against anything either intentionally malicious or just like accidentally wasteful in terms of computation or whatever it might be.

I think that there are a couple of places where it certainly could make sense to think about serverless in an IFTTT architecture in the future. I think sort of a lot of people's mental model, and certainly one that's attractive to us too is this idea that we just have this constant stream of data coming from the Internet and it had something like a Kafka or a Kinesis. Then from that we just execute Lambda functions or serverless functions against that stream of data.

[0:32:19.1] JM: Yeah.

[0:32:19.6] NL: I think that one of the challenges is that plagued the reality of the Internet today is that the real-time story just like isn't there for that to be the only way for us to get data. We need to have — The vast majority of the events that come into our system today are coming in through polling. The good news is that's changing. As time goes on, proportionally more events come in through real-time, and eventually we'll get to that point where real-time is sort of dominant. Until then, we need to invest heavily and infrastructure that allows us to do this constant polling.

Then sort of from a practical perspective, I think that you trade off a lot with Lambda in terms of your ability to have like certain types of instrumentation and introspectability. Things like the cold start time can really matter for us. You could imagine that the difference between telling your Alexa to turn on your lights, if that happens in one second, because that action hit a Lambda that was warm, versus three seconds, because it hit a Lambda that was cold. That sort of like difference in sort of your perception of the product can really add up.

There are obviously a lot of places where the timeliness matters less. If you're pending a line to an Evernote node. Most people aren't like staring at their phone waiting for that line to show up, but there are a lot sort of more IoT or interactivity use cases where latencies is like super, super important for us.

The other thing is that the sort of cost benefit of Lambda I think depends a lot on their being really sort of like periodic load on those functions. If you have like a Lambda function that's executing effectively 24/7, the cost of Lambda is going to be higher than EC2 and certainly higher than like a reserved instance or something like Spot.

Having like your full workload on Lambda for something like where you how have like a really pretty steady state of transactions that are happening, you might not come out ahead on a cost basis. There are other benefits too, like — It's less stuff that you're managing actively.

I think it's a lot of long trade-offs that I think a lot of people are of digging through. I'm frankly super excited about all the work that's happening in that space and I think that it will continue to mature as a set of products. Very well in the future, it may be sort of the only sensible way to build internet services, because the expertise and sort of like all the extra work that has to go in to actually like running servers, it kind of gets abstracted away in a really powerful way by Lambda, and we just need to o make sure that the tooling is there and the costs makes sense, things like that.

[SPONSOR MESSAGE]

[0:35:24.7] JM: You are programming a new service for your users, or you are hacking on a side project. Whatever you're building, you need to send email, and for sending email developers use SendGrid. SendGrid is the API for email trusted by developers.

Send transactional emails through the SendGrid API. Build marketing campaigns with a beautiful interface for crafting the perfect email. SendGrid is trusted by Uber, Airbnb and Spotify, but anyone can start for free and send 40,000 emails in their first month. After the first month, you can send 100 emails per day for free. Just go to sendgrid.com/sedaily to get started.

Your email is important. Make sure it gets delivered properly with SendGrid; a leading email platform. Get started with 40,000 emails your first month at sendgrid.com/sedaily. That's sendgrid.com/sedaily.

[INTERVIEW CONTINUED]

[0:36:39.3] JM: The show that we did a while ago about this startup where they solved a lot of their problems with burstiness in traffic by using AWS Lambda, that is not the problem that you have at IFTTT, because all day long you need to be polling these services, because you need to

be listening to a service like Twitter where they don't have some sidecar process that's running that saying, "Oh, we're listening for events that we need to send IFTTT. You need to listen to Twitter," and the only way to listen to it is to constantly poll for the information, like if you've got a hundred different users who have a hundred different applets that they've instantiated where they're using Twitter as something that they're plugging into, you need to be pinging Twitter all the time on like looking at those users accounts to see what they're tweeting so that you can make calls to your own internal infrastructure.

Because you've got this constant need to be scheduling polling requests to all kinds of different websites and APIs, you might as well just have a full-fledged servers up on a regular basis rather than going through the process of spinning up Lambda and spinning down Lambda containers and dealing with the cold start problem all the time, the latency and then just the issues from that start up, the costs of starting up and spinning down machines. You're not going to incur that if you're just requesting EC2 instances and having them just be long-running, because you're doing polling all the time. Interesting.

[0:38:23.6] NL: I think that's spot on. I would add that even if we just look at the data that comes to a real-time infrastructure, sort of like the stuff or the events that would be a perfect use case for something like Lambda, because we have users all over the world, there aren't sort of like what you'd expect to see as periodic changes in our throughput. We are surprisingly constant in terms of the amount of work that we're doing, which in some ways is really good, because it means that we're not being wasteful with our infrastructure by and large, but it also means that like the ability to do smart scheduling around like what compute resources do you have available at a given time. There's like really aren't that many opportunities for us to do that today. It's kind of nice to not have a problem to solve, but it's also kind of a bummer, because there aren't like forces pushing us to try out some of these more novel approaches.

[0:39:18.0] JM: How important is cost controls? Are your unit economics good enough the you can have a pretty laissez faire approach to pouring money into cloud services, or do you have to make certain build versus buy decision?

[0:39:33.7] NL: That's a really good question. I think that for us, we've gotten to a point where things, services like our outbound email service and our outbound SMS service account for

maybe not an equal amount, but certainly like approaching equal to our monthly cloud spend. In terms of like our ability to like to reduce our total cost, there are things that are sort of just like baseline expensive, because you were not going to operate telephoning services.

I think that, generally speaking, we do try to be very sensitive around our infrastructure costs though. About a year, a year and half ago, we went through a pretty intense process of trying to cut costs where possible, thinking about ways to optimize our services, thinking about things like the data that we're storing on S3. Are we storing that in ways that makes sense? Are we storing data that we're never going to retrieve again? Are we storing it compressed? What are the axis patterns, etc., etc.

We also had a bunch of services running on Heroku, because the deployment and like management story was so simple and we had a bunch of users who are like used to them. They're like, "Okay." At a certain point that made a lot of sense for us. Then we had invested enough in our Mesos infrastructure, the sort of incremental cost of running an individual container and an individual application was approaching basically zero. We're able to fold a lot or consolidate a lot of that sort of disparate compute into a single place where we have a lot more control over the macro view of cost, and that's helped a lot.

The really dirty secret about IFTTT is that our run time is all written in Ruby. What that means is blocking I/O, which is just like absolutely insanity if you think about what IFTTT has to do, which is make API calls all over the Internet, and that's all we do. We just do I/O. We are, like in an absolute sense, being super wasteful with our compute and that's something that we're actively working to change. The goal is to move to something, a different type of runtime that allows us to do non-blocking I/O, and that I think is going to have a tremendous impact on our bottom-line cost basis.

[0:41:57.3] JM: Are you thinking Go, Rust or Node?

[0:42:01.0] NL: Our sort of inclination right now is to the JVM, primarily because of library support. As heavy AWS users, like the Java APIs tend to be sort of the top tier, or SDKs tend to be the best. On the JVM — And there are a lot of like really cool things that you can do on the JVM. We have some folks who are comfortable with Java. We have some folks who are

comfortable with Scala. You have some opportunity to actually like have different experiences, but still have those core sets of APIs and SDKs being really, really solid.

[0:42:31.5] JM: I don't know how much you can go into this or if you're even familiar with it, but the whole economics discussion of running a business, like IFTTT, fascinates me. You mentioned something very interesting, which is that the email service — Kind of, I guess, email and SMS hooks into — I guess that's basically like if Domino's sends me a pizza, I want to send an email to my mom and tell her, "Oh, I wish I was eating your home-cooked food. I'm ordering pizza right now," that kind of service.

Yeah. That kind of service is accounting for a large percentage of your — That's accounting for maybe 50% of your cloud revenue, and then a similar service for —I'm sorry, your cloud costs. A similar service on SMS is accounting for — I don't know, another 50% or 40 and 60. Basically, those net out to the cost of your cloud services.

I'm actually curious about the overall economics of the business, like how much — How operationally intensive for example is your cloud service. Do you have a bunch of ops people that have to be on call all the time or is it more of a no-ops type of system, and how much of the cost of the overall business are headcount? Just like when you look at the macro of the overall business, is it working and what needs to change to make it work, of there are certain tailwinds in Internet usage trends that are going to make IFTTT an extremely profitable business in the near future? Maybe you could tell me about the economics near term and longer term and how that impacts engineering decisions.

[0:44:11.9] NL: Sure. Frankly, I have fairly limited visibility into our overall cost structure, but things like headcount obviously, I have a pretty good visibility into. Our engineering team is really pretty small. It's about 15 folks. Our infrastructure team is two.

[0:44:30.8] JM: That's great.

[0:44:31.4] NL: We've done a lot of work to make sure that we are able to sort of operationalize things in a way where you need to have the minimum amount of human intervention, and it's been really difficult, but also really rewarding.

[0:44:45.2] JM: Yeah, that's discipline. I got to say, that's really discipline. IFTTT has been around for a while, and to keep the engineering team at 17, that's very disciplined.

[0:44:55.3] NL: Yeah. We're obviously still trying to grow, but we're not at a point where it makes a lot of sense for us to double or triple or 10X the size of our engineering team to solve some of these problems yet. I think that, frankly, we still have a lot of opportunity to further optimize the way that we are applying our engineers to some of these problems.

In terms of sort of like our overall business model and strategy, as it stands today, obviously we're still a startup, if an old one. These things are subject to change, but we don't have any interest in charging our users for access to the service. What we do is we charge partners who integrate with IFTTT, basically like a yearly licensee fee.

At the very minimum, you have to pay anything — I can't recall. I think it's like $200 a month to be a published partner on our platform, and then from there we offer sort of incremental services, whether that's like white glove handling of your — Building your service. We offer professional services. We offer marketing, etc. The ability for someone to have like a very light engagement with us or really go deep in terms of how embedded they want that experience to be.

[0:46:11.4] JM: It so interesting thinking about how IFTTT, the prospects for the future, because you can imagine two most equally plausible futures where IFTTT becomes super important, or gets obviated by Google or Amazon or somebody else, or just the relationships between the giants changing. For example, if Google and Amazon and Instacart all made it easier to plug directly into each other, and Google's got there IoT platform and their Google voice assistant and stuff and they just make it more open, then everything would be copacetic. You could just wire your Google home into your Amazon Alexa and your Instacart and they would just ignore their competitive frictions and play nicer together. But you look at somebody like Andy Rubin who, with Essential, is trying to make a hub that connects these two, these different businesses that have competitive frictions, because the competitive frictions between them end up being frictions for the user. Somebody like Essential tries to bridge the gap between those different companies, I see IFTTT as like a middleware that is kind of doing the same sort of thing that

takes the same approaches Andy Rubin with Essential and saying, “Let’s make a way to bridge these services together in a way that is agnostic of their competitive frictions.”

I don’t know. Are these similar thoughts as to what goes through your head?

[0:47:51.5] NL: Absolutely. I think that we have long seen the value and us remaining as much possible a neutral party in sort of this online landscape. I think that in the situation where Google and Amazon and Instacart and whoever, the giants as you say, they had together and they say that they’re going to be cooperative in terms of their integrations and their APIs etc. In a lot of ways that would solve a number of problems.

I think that another place where it provides a lot of value is putting smaller companies, smaller services, smaller products on an equal footing. This is more like my personal take, necessarily the company view. If all that happened was that Google, Amazon, Apple, whoever, decided that they were going to work together and have like those products work really well together, it’s kind of a net lost, because it really closes the door on a lot of interesting innovation that might happen at a smaller scale.

I would really bummed if there were less interesting IoT products or like cool social services launching on a regular basis, because there was as much larger walled garden that served a lot of users pretty well but doesn't necessarily solve all the use cases.

[0:49:11.2] JM: Okay. We jumped to the business section of my preparation. Let’s scale back. I know we’ve only got 10 minutes or so left, but I want to go through the rest of the infrastructure or at least some of it. We kind of talked end-to-end how a request from Domino’s is going to translate to my Phillips Hue light bulbs being turned on. We talked about the server infrastructure that manages that contract between my different services that are bound together by an applet.

Let's quickly run through monitoring and data infrastructure so that we know how you are able to have visibility into this system and the analytics that come out of it.

[0:50:02.4] NL: Sure. I think that there are sort of three flavors of monitoring visibilities introspection to however you might want to think about it. We use typical APM tools, like New Relic, for our sort of like edge applications. Whether that's our APIs or our web servers, things like that are pretty familiar I think for most listeners.

We also use your typical sort of like Statsd comparable, time series, visualizations. We use InfluxDB and Grafana to sort of do aggregates and visualization on more like application-specific metrics and infrastructure level metrics as well. Things like present utilization of CPU across the cluster or 99 transaction time for a specific request hitting our load balancer.

Then from the IFTTT runtime side of things, we have really verbose rich output that goes to our — Like the final destination as our analytics pipeline, but we tap into that to throw it into Elasticsearch, and we do like regular aggregations and visualizations on top of that data as well to get a sense of changes in behavior, being able to look at things like changes in service time by individual service that we integrate with, things like that.

[0:51:29.0] JM: As I was reading about this the data infrastructure ,I read about some things like Elasticsearch, Apache Spark, some sections of the infrastructure where data goes in in batches, other places where it streams. Talk about some of the tools that you use in your data infrastructure.

[0:51:50.1] NL: Yeah. The very end of it, like our two final repositories for data are S3 and Redshift, and Redshift is sort of like the thing that most people are using to run queries against their data on a regular basis. Whether that's engineering trying to gather insights into like how people are using their product, or marketing team trying to do cross sections on users and behaviors and things like that.

Our data pipeline is a mix of streaming in batch using Spark for streaming and then AWS data pipelines for batch floating. Doing things like taking nightly snapshots of some of the tables in our production database and copying them into Redshift.

We also have some more specialized data products that sit on top of our streaming tools, like Amazon Kinesis to do things like predictive modeling for how often we should be pulling an API on an applet-by-applet basis, things like that.

[0:52:51.2] JM: When I worked at Amazon and also just hearing other people in the industry talk about what it's like to be on call at AWS. It sounds really hard. IFTTT is kind of an infrastructure provider. There are people who have built fundamental components of their business that sit on IFTTT, I assume. What's the ops strategy at IFTTT? What kind of SLAs do you try to have and how sensitive do you feel like the infrastructure is?

[0:53:23.2] NL: That' a really good kind of nuanced question, I think. In general, we treat our systems as critical. We don't understand the specific use cases that any of our partners or users are using itfor, so we kind of have to assume that they're all really, really important. I know that one of the really inspiring uses of IFTTT is someone builds an integration that allowed them to sort of do passive insulin monitoring for someone in their family. IFTTT doesn't actually like —

[0:53:54.4] JM: Passive what?

[0:53:55.3] NL: Like blood sugar level monitoring.

[0:53:58.8] JM: Oh, insulin monitoring. Okay.

[0:54:00.7] NL: IFTTT isn't sort of responsible for providing like dosing, but reminder as to like, "Hey, you should probably have a little insulin." Things like that. It's like, "We really need to have our systems performing reliably and in a fast perform way." Really important use cases.

I think that like at a fundamental level, it's about the same sort of engineering discipline that lets us get away with having such a small team. Making sure that the systems that we build are robust, and when things go wrong we know as soon as possible. Making sure that we have the right metrics in place to understand like what types of issues are problematic. Things like that. As well as having systems that are relatively adaptable at sort of routing around different types of problems.

When Twitter's API goes down or Domino's API gets slow for whatever reason, we have protections in place to make sure that that doesn't have downstream effects across all of our users, all of the integrations to make sure that things that continue to work continue to work really well.

One of the sort of like facts of running a business that is connecting all parts of the internet is that some things are always broken. Sometimes it's our stuff, but often times just like by the law of number, it's someone else's integration that's going to problems. We provide our partners with in-depth analytics around their integrations. We have alerts that they can subscribe to to know if there are issues with their APIs, things like that. Then we have pretty well-defined roles for different people who are on call.

Our infrastructure team is responsible for the base infrastructure. Anything that's sort of like at the control plane or load-balancing level they're on top of. We have a team that's on call for the IFTTT runtime, making sure that the systems, like the thing that in cue checks for applets is running and that we have enough capacity to execute all of the applets in an expedient way. Then our web team is on call for things like the website being down or reaching a critical level of errors. That really allows the people who receive a page to be able to act on it pretty immediately, because it's a system that they're intimately familiar with. It also creates positive feedback cycles, so like if you get woken up because of an issue in the code that you deployed earlier or that you had a hand in creating, you're much more empowered to go in and like understand what that was, figure out how to make a change and help yourself not be alerted by that in the future.

[0:56:38.3] JM: All right. This has been a really interesting conversation, Nicky. Maybe we could close off, tell us what IFTTT is working on right now, maybe some of the big engineering challenges that you personally are focused on.

[0:56:52.0] NL: I think that my focus at IFTTT is on the runtime, and as a team we are really focused on what I hinted at earlier; moving our core Ruby runtime to something more perform, probably on the JVM.

[0:57:09.8] JM: What kind of cost savings are you going to get out of that?

[0:57:12.9] NL: It's really difficult to say. I would guess like for our part of the system, probably on the order of like 10 to 20X. But it's easy to be like really optimistic about that. It's also hard to underestimate the total cost of blocking, again, in a system like ours. Sort of like went depending on how we're measuring it, we are at sort of like 90% to 99% idle at any given time. There's a lot of opportunity to make some big improvements there.

[0:57:46.2] JM: All right.

[0:57:47.4] NL: Yeah, that's it.

[0:57:48.4] JM: Cool. Anything else you want to add?

[0:57:50.1] NL: I don't know. We're hiring. If folks are in the San Francisco Bay area interested in any of the projects that we're working on, whether it's our runtime, our infrastructure, our data products, our mobile apps, our web products, we'd love to talk to you.

[0:58:06.5] JM: All right, and you can send Nicky an email perhaps, or send me an email. I'll connect you to Nicky, or you could probably find them on Twitter. I assume you're on Twitter.

[0:58:16.2] NL: I am.

[0:58:16.8] JM: All right. Great. Cool. This has been a great conversation. Thanks for coming on the show.

[0:58:20.8] NL: Thanks, Jeff. It was great.

[END OF INTERVIEW]

[0:58:25.1] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That's symphono.com/sedaily.

Thanks to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver this content to the listeners on a regular basis.

[END]