EPISODE 701

[INTRODUCTION]

[0:00:00.3] JM: A smart security camera takes in a high volume of video images and processes those images using a set of machine learning models. These models can be used to identify interesting snippets of movement throughout the day wherever the security camera is located. The security camera can decide which of those snippets to keep. Some of the video snippets might contain movement of birds, but other video snippets might contain footage of intruders, or people stealing merchandise from your store.

As the video stream is processed on the smart security camera and machine learning models are used to classify the entities in the video stream, some of the data gets thrown out as being useless. Some of the data gets sent to the cloud for additional processing. Some of the data might trigger an alert that there is an intruder on the premises, maybe that alert emits from the cloud, or maybe it emits directly from the local network that the smart security camera is on.

Each piece of video data is an event. These events are processed and acted upon. Modern applications are highly interactive and they have lots of events. Other examples of event data streams are website traffic data, self-driving car data, time series logging data, video game session data. Building applications that respond to these high volumes of events requires us to program triggers to react to data streams.

We need actions to take in response to the data streams and we need workflows to orchestrate what the overall picture of our application is doing as this application is consuming a larger data stream. Plus, we don't want to keep all of this data. We probably want to throw some of it out. Depending on the application, we may want to throw a lot of it out, we may want to refine the data. There's a lot of things you want to do to these high-volume data streams in order to operate efficiently over them.

Flogo is an event-driven ecosystem for building applications around streams of events. Leon Stigter and Matt Ellis work on Flogo at TIBCO, and they joined the show to event-driven application development and their work on Flogo. They also talked about the constraints of machine learning applications at the edge and how event processing systems like Flogo can be used to handle large data streams on edge devices.

Full-disclosure, TIBCO is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:02:42.6] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily. Thanks to DigitalOcean for being a sponsor.

The co-founder of DigitalOcean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW]

[0:04:49.6] JM: Leon Stigter and Matt Ellis, you both work at TIBCO. Guys, welcome to Software Engineering Daily.

[0:04:54.8] LS: Hey thanks.

[0:04:55.6] ME: Thank you very much.

[0:04:56.6] JM: Today, we're going to talk about Flogo, which is an event-driven ecosystem. It's an ecosystem for building applications that have events, or they might have lots of stream processing going on, and we'll get into what those different facets of the application ecosystem mean. I want to start with one specific application, which is machine learning on edge devices.

This is an interesting and burgeoning area, because we have edge devices that are ingesting large streams of data, for example of video feed. The edge device can facilitate machine learning across that high throughput data stream, but it's obviously a challenge because you're bottlenecked by the amount of storage, the amount of networking capacity on this edge device, and so you have this trade-off between what are you doing at the edge and what do you want to try to do in the cloud? If we're talking about specifically machine learning on edge devices, what are some applications of machine learning for edge devices?

[0:06:02.2] LS: Yeah, that's a fantastic question. I mean, if you truly think about it and you've called this out already, you've got these edge devices. I mean, you've got thousands of them potentially that are producing data points and you could be producing data points roughly a data point per millisecond, or even at higher frequencies.

Oftentimes, taking that that individual raw data and transporting that back up to the cloud is not practical. Typically, what you end up doing is you apply some element of stream processing. What I like to say is that in this case, stream processing is oftentimes used as data pre-processing. If we take a concrete example, we've demonstrated the idea of taking an edge device with an accelerometer, and the accelerometer is used to predict whether or not somebody's walking, running, maybe they fall in the – how great, or what was the magnitude of the impact and things like that?

You process the stream of data and then you prepare the ML feature input into a format that's compatible with the ML model that you're inferencing, and then you can leverage things like Tensorflow to inference the raw data, or the processed data and then come up with a classification. Being able to classify that the based on accelerometer readings that the person's walking, running, they fall in and then you've calculated the magnitude, so the impact was rather great. Maybe we should alert somebody for some help, or something like that, is an ideal scenario where you want to perform the logic on the edge device, because –

I mean, you can imagine. If you were streaming up that data up to the cloud and it's producing 3,000 data points per second, obviously the person, especially if their internet connectivity was flaky, as is the case with most edge devices, then the person could theoretically be – they could be quite hurt before you figure out that they've actually hurt themselves. That's one concrete example of where and when you'd want to start performing ML at the edge.

[0:07:57.1] JM: Right. This data pre-processing, or data cleaning that you can do on the edge device. I've also heard the example of – for example, if you have somebody that's moving around in space and their phone is recording lat/longs and the lat/long will indicate where they are in space, but if the machine learning algorithm that you're going to feed that location data into only reads, for example the restaurant that you're closest to, then it might be useful to interpret that the lat/long into the closest restaurant in some given geo space and you might just have some small subset of geo-locations correlated with restaurants that sit on the device and allow you to very quickly transform lat/long, raw lat/long data into restaurant data, and then you can feed it into the ML model, have the ML model do its own interpretation and transformation, and then maybe you send it up to the cloud, or you act on it locally.

[0:09:01.5] LS: Yeah, exactly. That's another good use case.

[0:09:04.5] JM: Right. You often are going to do this pre-processing. Is it also useful to think about the workflow of what you're going to be sending between the edge devices in the cloud? Is that another application that you might use Flogo for? Perhaps once you send this pre-processed data through the ML model, then you maybe have some other stage in your workflow where you determine am I going to send this to the cloud?

[0:09:32.8] ME: Yeah, exactly. That brings up a really interesting point. If you think about how ML models are produced, typically speaking supervised machine learning models are produced by a data scientist, or data analysts, or somebody analyzing the data, coming up with general assumptions and then testing out their assumptions. Eventually building some ML model that can then be run. The point is they've got a set of data that's labeled, that they can then use to train models against.

There's a couple areas where you'd want to send data from an edge device up to the cloud. The first one and this is let's assume the model has already been built and it's already pretty solid. In this case, then what you'd often do is as you'd end up sending the resulting action up to the cloud. You do the data prep, but the streaming pipeline within Flogo, you'd inference the ML model within Flogo as well and then you'd send the potential result, or the output of the ML model to the cloud. You're not sending in those 3,000 data points per millisecond. Maybe you've aggregated it up per half second and you're only sending one data point every half second.

Then the other side of the coin is how do you make the model better? To do that, it's essentially an infinite loop where you constantly collect data, you're constantly retraining models and things like that. In reality with the current set of tiny edge devices, you're definitely not training models at the edge, so you still need to send that data up to the cloud.

We're working on actually a couple exciting things in Flogo that we're calling – at least at the moment, we've codenamed it the metric service that that'll out-of-band persist, as much as possible based on storage on the device. Persist this data and when a reliable network connection is available, it'll send that up to the cloud to allow the data scientists to continue getting that new set of data, and then deliver updated models, maybe kick off champion challenge, or bake off some device and things like that and let two models execute and you can compare the results and things like that. There's a lot of things that still need to happen. Executing and taking action is obviously something that's massively important, but that's only one portion of the equation.

[0:11:40.5] JM: Flogo is described as this event-driven ecosystem. It's not exactly a framework. Describe what the goals of the Flogo ecosystem are and what the scope of the project is.

[0:11:54.3] LS: Yes. This is rather interesting. If we go back in time two years ago or so, we sat down and we said, “Hey listen. You've got this this new emergence of compute that's coming online. How do you build and process data at those edge devices, but also how do you do it in a cloud native way and how do you build this concept or this framework?”

We took a step back and we came to the realization quite quickly that you cannot simply take an existing tech stack that's maybe built in Java, or Nodejs or something like that and retrofit it to work on an edge device, or a device with much less compute. We began building out Flogo. We picked Golang as the language of choice for a number of reasons. It compiles down natively, it compiles quite quickly, the binaries are statically produced, or statically compiled, meaning that you've got no third party dependencies that you need to pull in at runtime. The only thing you really need on these edge devices is an OS, which in this case is oftentimes Linux. Then you've got this tiny binary that runs.

It started out in that capacity. IoT was a classic example, because you had resource restricted environments. Though cloud native was also something that was in the front of our mind as we were building out the framework. Then time progressed and we said, “You know what? Listen.” We've built this framework that is event-driven by design. Meaning that you take an event, you dispatch an event to a worker, a worker does something with that event and that's it, right?

We took a step back and said, “Hey, listen.” An event is fundamentally just something that happens. There can be one of them, there can be millions of them. It's how you process the event that's actually different. We began exploring the concept of stream processing as it relates to ML model inferencing. We quickly came to the realization that hey, you know what? Oftentimes, this raw data is actually meaningless on its own. How do you produce a derived value that's meaningful and allows us to execute predictive models?

We implemented the stream processing action, or in other words the stream processing event processor and that's how it started expanding into an ecosystem, because of the way it was originally built from the beginning that it does process events. We can add in different event processing capabilities, expose those out to this ecosystem notion and then developers have a multitude of ways that they can build applications. They can build applications using a web-based environment, or they could use our Golang lib and write their own Go applications and they can use it in a number of different capacities. That's how it expanded over time. Hope that makes sense.

[0:14:42.9] JM: Why has there been an increase in people developing event-driven applications?

[0:14:49.8] LS: It's a good question. I mean, in reality there's nothing new about event-driven applications in general, right? I mean, we've always had this notion of an event and how events are produced and consumed has a –

[0:15:02.2] JM: A change in the nomenclature, perhaps.

[0:15:04.3] LS: Yeah, exactly, exactly. Also technologies evolve. We've moved into things like MQTT and we've moved into consuming, or using things like Kafka. The notion and the simplicity of leveraging those technologies has brought back up the importance of an event and less so of the traditional request reply pipeline that we've been used to for quite some time.

[0:15:29.5] JM: Let's talk about the vocabulary within Flogo. You have triggers, actions, handlers, flows. Explain what a trigger is and an example of a trigger.

[0:15:42.1] LS: Yeah. A trigger is that event sync, or event consumer, if you will. Something like a Kafka consumer, or an MQTT listener, that's the notion of a trigger. That's the instance, or the go routine that's sitting there listening for events to come in on that particular pipeline. That's what a trigger is.

[0:16:05.8] JM: What would be an example of that? If I'm developing my Flogo application, or my Flogo microservice, what would be an example of a trigger?

[0:16:14.9] LS: Okay. In this case, if you're developing an application you look at it holistically in the sense that okay, so I want to consume events from some source. I want to consume events from Kafka and I want to consume events from MQTT. Kafka and MQTT are both my triggers, okay? Those are listening on a certain Kafka topic, MQTT topic, things like that, waiting for events to come in.

Once the event arrives at the trigger, the trigger dispatches that event to a handler and the whole purpose in life for a handler is essentially to decide who processes that event, or in other words what action processes that event. Then a handler will dispatch that event to one or more actions and then an action processes the event in a capability that's relevant with that particular style of event processing.

If you're processing a singular event and just maybe you're enriching the event and writing it to a data store, that's a flow, okay? A flow is a process. It is essentially a process engine that includes things like flow control and branching and conditional logic that allows you to take an event and process that single event and then continue processing events as they come in. If you wanted to do something like a stream processing, then you pass it to a stream action and the stream action has this notion of statefulness that persists over a period of time and maybe you're aggregating something over a period of time and things like that.

You've also got the notion of an internal channel, so all of these individual event processing capabilities or actions can pass data from one point to another. Maybe you initially have a trigger handler dispatches to a stream pipeline, a stream pipeline aggregates over a period of maybe 50, or whatever, a 50-second window, does some other aggregation, does some filtering, and then passes processing over to a flow, which is as I said is a process engine. That allows us to do things like check conditional logic, write things to a database, maybe we enrich the data and then we maybe we publish that back out to another Kafka topic.

[SPONSOR MESSAGE]

[0:18:30.0] 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, CICD, 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-prem 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. I remember people saying that this is a platform for building platforms. 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 platform as a service on top of, which is why OpenShift came into manifestation. You can check it out by going to softwareengineeringdaily.com/redhat and find out about OpenShift.

[INTERVIEW CONTINUED]

[0:20:38.7] JM: Let's say I'm using Flogo to handle a video stream that's sitting on a security camera, a very smart security camera and the smart security camera is on a shipping container intake place. Let's say this is a – during the day it's very active, there's lots of people moving around and they're entering and exiting this building near this security camera, but there's also some birds that are landing there. Then during the night there should not be anybody wandering around. There's a bunch of floodlights. It's a well-lit area, but for bird flies through in the middle of the night, you don't want to set off the – you don't want to send an alert to a security guard.

If a person, like some shipping container bandit is running through the shipping container yard, you do want to be able to process that a person just ran through the yard. You've got all kinds of machine learning models for detecting these kinds of entities. Would this be a good use case example for us to give an example for how triggers and actions and handlers and flows might be written for this edge device?

[0:21:46.1] ME: Yeah, exactly. I mean, in this case the camera is almost similar in the sense to that accelerometer example that I had used before. In this case, you'd have the binary stream of a video streaming over to a Flogo trigger. The trigger would then de-serialize the stream, convert it into pictures. Maybe you're skipping certain times, so if it's a 30-frame per second camera, you obviously don't need it probably, maybe, I don't know, but you don't need to look at all 30 frames per second. Maybe you only do one frame a second, or something like that.

Then you pass that over to a flow, which could then pass that that image, the binary image representation over to something like a Tensorflow CNN model that could then output the classification of the model. Whereas, maybe it's all good or hey, there's some abnormal movement and the resulting output of that inference could then – Flogo could then send a message, SMS, or some other paradigm to another system to alert security.

[0:22:50.1] JM: Would the output of the model go through an action, or through a handler, or its own flow? Would you define an additional flow for the output of the model?

[0:22:59.4] ME: In this case, it would be within the same flow. The model is executed as what we call an activity. An activity is essentially the unit of work that's performed. Execute ML model, or inference Tensorflow model. The output of that would be published onto the same flow, the flow process, or the flow scope that's currently being executed. It would be part of the same flow.

[0:23:23.6] JM: The way that many people are writing this code today is not within a well-defined framework, or not within a well-defined ecosystem. Maybe they're just writing random Nodejs services that are sitting on this security camera, and eventually the service is going to ping the Tensorflow model that's sitting on the camera as well, and then just makes a network call. What's the advantage of having this framework, or ecosystem of well-defined vocabulary and well-formed vocabulary of defining these flows?

[0:24:00.5] ME: Yes. I'll let Leon jump in as well, but quickly, so if you take a step back and look at look at this. When a developer is – I mean, developers will want to build applications in the language, or using the framework, or lib, or whatever that they're most comfortable with. What we've tried to do with Flogo is capture the notion of event-driven processing using the standardized terminology and constructs and things like that as we said. I could build an application for an edge device, I can build an application for a serverless function, or for a containerized deployment and I could do that using Flogo.

I don't have to change the tech that I'm using. It's all baked into the ecosystem and framework. I can also leverage, there's over 400 different community contributions that exist today. I can leverage those community contributions and get a head start as to how I want to maybe process events and things like that. If you take Tensorflow inferencing as a solid example, there's been a fair bit of work that's been put into the inferencing capabilities within Flogo, which is definitely not a complicated thing, but not an easy thing either, right?

I mean, it's not something that a developer would necessarily want to do from the ground up every time, or build their own in Nodejs or something like that. They got to leverage a lot of the capabilities of the entire ecosystem to fundamentally get their work done quicker. We also provide the Go API. If you do like to code and you don't want to do some graphical representation of your application logic, then you could absolutely use the Go API and then still write your own code and then just bring in the libs that are relevant to you.

[0:25:38.2] LS: I would almost echo what you said. In fact, I was talking to a group of developers today, just now actually before this recording. We were walking through building some back-end APIs that ultimately had to deployed as containers onto their container platform. They were very surprised about how fast they were able to do everything using Flogo, as opposed to what you're doing through Nodejs.

Then one of the amazing benefits that they saw within five minutes is that you can build your app, you can design everything through that web UI that Matt was just talking about, and then test it locally, or through that web UI running on for example, the Windows machine. Then without changing any code, just deploying it into a container and take that container, run it in obviously a first test and then acceptance test, that good stuff. Ultimately, taking that binary without changing any code and just running it as a container on a large platform.

For them, the added bonus of being able to take and design the application graphically, being able to take that metadata that describes it, put it into a version control system and ultimately have a clear defined CICD pipeline that then builds and tests and ultimately creates the Docker container for them to deploy, for them that was the amazing benefit they got from looking at Flogo.

[0:27:08.3] JM: Right. There, you mentioned the web UI that you can use to visualize these flows that you're developing. You can have a visualization, a web UI for actually understanding the security camera example. You could imagine that getting very complicated very quickly. You've got this raw video stream feed and then it does this stuff during the day, it does this stuff during the night and here's what it does if it sees a bird, here's what it does if it sees a human. That could get very complicated. Having a web UI and a visualization, you do have these nice little visualizations. It's like UML diagrams that let you understand what's going on, rather than having to dive into the code and look through switch statements and if statements. I can imagine that would be pretty useful.

[0:27:52.9] LS: True. I mean, I think if you look at what people tend to use most, it's essentially starting with the web UI, simply because it gives them a graphical way of representing their flows, and obviously that helps them as you said as well, visualizing whether it is that they're building, whether it's a small micro-service that they ultimately end up deploying to things like AWS lambda. Or in the example that you set, where they take a stream of videos, run a bunch of machine learning algorithms on that, and take a decision what to do. It's an easy way of grasping what it is that your flow was doing.

The other thing that at least I have seen talking and talking to developers is that a lot of people actually enjoy writing code, and I think that in today's world if you can – if you can write proper code, that's actually a great thing to be able to do. The thing I absolutely love about Flogo in that context is as Matt was talking about, the amazing set of contributions the Flogo community has built, I can actually take those contributions and use them in my – in this case, Golang code, so that I don't have to think about creating a database connection, or writing to a file, or even inferencing a machine learning model, because that is not something I write on a day-to-day basis. I can very easily take that Flogo activity and use it in the exact same way that someone would do from within that web UI.

[0:29:25.2] JM: This ecosystem is focused around Golang. What are the advantages of using Golang for the event-driven application development?

[0:29:34.4] ME: Good question. Go is obviously a pretty performant language. It's efficient. It's also efficient from a footprint perspective. Number one, you've got this notion of when you build an application you've got a statically compiled binary. You have no additional third-party libs that you needed to deploy on your OS. You don't need to deploy an additional framework, or something like that. You don't need the JRE, or Nodejs installed. There's literally nothing else, other than your binary and the operating system. That in itself is massively attractive.

Also, the Go language itself is becoming very, very popular. The notions, or the preconceived notions that I think the Go community has is something that might frustrate a lot of OP devs, but at the same time is simple to consume and read. If you're also reading the code and wanting to contribute to the project overall, it also becomes something that is quite honestly very, very nice to consume and understand as well.

I mean, they've got notions, like if you have to – if somebody can't understand your code within the first few minutes, then you've probably overcomplicated it for no absolute reason, so you should look at simplifying and streamlining and things like that. I think that's one of the notions and the concepts that I really enjoy about Go.

[0:30:49.7] JM: Flogo is as I understand, a single node stream processing system. If there's people who are out there and are thinking, "Okay, this is yet another stream processing thing. We've got Spark, we've got Flink, we've got Kafka streams. How many of these things am I going to have to learn?" Flogo is a little bit different, because it's not like you're architecting a distributed system of stream processing. Is that accurate?

[0:31:14.0] LS: Yeah. That's a good way to put it. I haven't really thought about it from that perspective. Yeah. Essentially, it's definitely not – it's not Spark. You're not trying to do some massively distributed batch-based stream processing. In this context, Flogo, or at least the way we've built it was specifically designed to consume from a couple event sources and then aggregate short-lived aggregations over a period of time and use that to pre-process real-time data. Not data that's sitting in a data store or anything like that. It's purpose-built to process real-time data that's being produced.

Then oftentimes as I said earlier, the stream is almost doing a data pre-processing for something more. Maybe you just want to pre-process the data and prep it to write it to some reference data store, or maybe you want to pre-process it for ML inferencing. Again, it's that real-time aspect that's slightly different from something like Spark.

[0:32:11.7] JM: Yeah. I mean, which is useful. Having a framework for doing processing on a single node and having a well-defined way to do that. You don't necessarily need to have everything be distributed across 50 nodes and ingesting gigantic multi-node pipelines, like a Kafka cluster.

In addition to stream processing or doing this security camera example, you can also write microservices, I guess? Or are these microservices specifically to do with stream processing? How would that differ from people just writing microservices in Golang or Nodejs?

[0:32:51.2] LS: Yeah. That's a good question. The notion again around Flogo is that it's event-driven, it's an event-driven ecosystem. It's not just for stream processing. Stream processing is just one of many ways that you can process events. If you want to process singular events and deploy this in the context of a microservice, that's also something that's absolutely doable. You'd use the action implementation, which is the flow action implementation to model your execution graph, if you will. Model how your application, or your flow will process data.

An event comes in off of a rest trigger, or a Kafka trigger, or whatever, right? Maybe you enhance the data, you query data from another data source, you enrich it then you write it to a database and you do some other logic, and then you return the result back to the caller. That's how you'd start implementing your micro-service.

Then again, we have that Go API, so that you could leverage those 400 contributions that exist in the Github community, to then code your API, or code your micro-service just using Go. We've got this other concept of – we've talked about the triggers already, but one of the unique benefits that we haven't yet touched on is that you could actually have multiple triggers bound to a single action.

If this case, if we use the action as the flow, you can have multiple triggers bound to a single flow, okay? The flow itself is almost treated like a function. We like to say it's a function as a flow, if you will. The flow has input parameters and the flow has output parameters. When the event source, or when the data comes in off of one of these events sources, the trigger or the handler actually maps the trigger data to your flow, and then your flow operates against a predefined schema, or a predefined object set, so the logic in your flow doesn't actually have to change even though you're consuming data from multiple different triggers. In this case, this allows you to really build robust and flexible microservices in the sense that you can consume data from multiple different sources without changing any of your app logic.

[0:35:00.7] JM: If there are people listening right now and they're still trying to figure out whether Flogo fits some application use case that they have, can you help them think through this? Who should be using Flogo?

[0:35:13.3] LS: Yeah. That's a fantastic question. I think there are a number of different personas that should be looking at Flogo. First and foremost, the most generic sense I'll say, anybody that wants to build event-driven applications should look at Flogo. How do you want to process these events? Are you processing singular events? Are you processing a stream of events, or maybe as one of the other actions that we haven't talked about is Flogo rules, which is a at the moment it's a read implementation of a rules, that rule network that can perform execution of rules against known facts.

It is stateful in the sense that historical facts can be persisted in in-memory and then you can execute rules against incoming events, and then join them against those historical facts. I'd say generically, if you want to look at – if you want to get into event processing, you want to – you're not quite sure, I'm going to consume from Kafka today, but that could change later, but I don't want to change it, or redo a bunch of application logic, definitely look at Flogo.

If you want something that's lightweight and efficient, absolutely look at Flogo. If event-driven is something of interest to you, you need something that's lightweight efficient and you need something that has machine learning constructs built as native first-class citizens within the overarching ecosystem and framework, then absolutely look at Flogo there as well.

[0:36:29.6] JM: What are some other example applications that you have used Flogo for?

[0:36:35.1] ME: This one I will kick over to Leon. He's built and then help the people build quite a few different apps from like Slack integration to a bunch of other stuff. Leon did you want to?

[0:36:44.4] LS: Sure. I'm actually starting off with the Slack integration, simply because that seems very popular nowadays. Actually, the group I was talking to earlier today, it was one of the things that they wanted to build, simply because it's an easy way to extend Slack with even more capabilities than you already have. As Matt was talking about the event-driven paradigm, and in this case it would be sending a Slack command, or just a Slack bot listening in to everything that is going on in a particular channel.

Some of the other applications that I've helped build is also very simply people automating their day-to-day jobs. Whether it is grabbing data from a bunch of places and then presenting it in a way that they can either send to their manager, or send an e-mail or something, but probably the most interesting thing that at least I have heard about from a person just building a home automation, what he did is e took Project Flogo, put it on to a smart doorbell and basically lets his doorbell decide whether or not to open the door based on the person standing in front of that doorbell.

Basically, the use case there is that you ring the bell, the doorbell sees who you are, takes a snapshot, if you will, then compares that to everyone it should let in. If it matches, it literally opens the door for you. That to me was an amazing use case as that guy was talking about it.

[SPONSOR MESSAGE]

[0:38:23.0] JM: We are running an experiment to find out if Software Engineering Daily listeners are above average engineers. At triplebyte.com/sedaily you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics; general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast.

I will admit that, though I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself. If you want to take that quiz yourself, you can help us gather data and take that quiz at triplybyte.com/sedaily.

We have been running this experiment for a few weeks and I'm happy to report that Software Engineering Daily listeners are absolutely crushing it so far. Triplebyte has told me that everyone who has taken the test on average is three times more likely to be in their top bracket of quiz scores.

If you're looking for a job, Triplebyte is a great place to start your search, it fast-tracks you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time, which is what I try to do with Software Engineering Daily myself. I recommend checking out triplebyte.com/sedaily. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8-bytes.

Thanks to Triplebyte for being a sponsor of Software Engineering Daily. We appreciate it.

[INTERVIEW CONTINUED]

[0:40:21.2] JM: Talk more about other services and tools in the ecosystem that people might be using Flogo with. For example, I think of things like AWS lambda, or perhaps any number of these APIs for machine learning, like image recognition, or object classification, or audio transcription. How do you see Flogo fitting into that increasing serverless application ecosystem?

[0:40:53.9] LS: It's funny that you ask about serverless. We were actually the – at least the first application integration vendor that launched native lambda support for Flogo. Remember that notion a bit earlier where I said that a Flogo flow is essentially a function. Essentially, what happens is you can leverage this notion of a lambda trigger. The lambda trigger when you build your application, you essentially every flow that has a lambda trigger gets produced as its own function that can then run on AWS lambda.

We've even seen people use the lambda trigger to write Flogo applications that actually serve up Alexa actions and things like that, so they've bound it to their echo and then that you can ask to do certain things and then it ends up hitting a Flogo flow that's running in lambda as a function. Any of the logic, though keep in mind not all of the event-driven capabilities make sense as lambda functions. Flows clearly do, because they're this singular event processing process engine, if you will.

Say a contextual rules engine doesn't really make all that much sense as a lambda function, because hey, once the function stops, you have to assume that any state that you're persisting is now gone. We definitely have things for lambda. We're also looking at Azure function support, as well as Google Cloud function support as well.

Serverless is something that we truly believe as being something that's quite important for developers and ops, ops guys in the future. We're really supporting that and then also the event-driven notion of serverless fits very well within our event-driven pattern also.

[0:42:39.0] JM: Flogo came out of TIBCO where you guys work. Why is this being built at TIBCO? What were the customer problems, or the just ecosystem changes that you were seeing that led you to building Flogo?

[0:42:53.5] LS: Yeah, that's a fantastic question. Yeah, a couple of things; obviously, we decided that we needed to evaluate how proper cloud native and edge compute devices would function in the future and how we build technologies that support deployment paradigms and devices. That was one angle.

We also firmly believe that open source is the right solution. If you take a step back, I mean, we fundamentally believe open source is the tip of the spear from an innovation perspective. Innovation will happen in the community, it'll happen openly and we'll mature the product as a community, not so much as a singular company. That's also a pretty big shift from our historical strategy, which is something that I'm quite proud of and really proud to be part of.

It's exciting, so yeah. I guess, the underlying point to your question was this is fascinating that it's coming out of TIBCO, but it's a new TIBCO where open source is the tip of the innovation spear.

[0:43:53.8] JM: I remember talking to a friend of mine who spent five or 10 years working in the "IoT industry," and this was – I think he worked in it early 2000s, maybe 2000 to 2010 around then, or 2015 roughly. IoT is something that it's been around the corner for 10 or 15 years, but here I think we're actually seeing it happen. It is coming to fruition and we're seeing that – I think you see that in the rebranding of IoT to edge computing. It's almost a more palatable reframing of it, because the word IoT has such a blemish around it. Do you think IoT is finally happening?

[0:44:41.7] ME: I think it's happening. It's going to continue to happen. I think the "IoT" segment is massively fragmented, massively confusing in the sense that it's not a singular Internet of Things in that form, so I do agree with you that rebranding to edge compute helps clean that up a little bit, but also changes the notion a little bit, right? It's about moving compute to the edge. Not just having a network of things, but it's about doing things at the edge.

I think that's where a lot of the new notions are starting to come up. It's what processing and what logic can I move to the edge, and then fundamentally reduce my cloud ingress costs, right? I mean, if you stick everything in the cloud and that's the central point for everything, then this network, this network of things, the IoT in the IoT realm has to communicate with the cloud for everything. I mean, that's massively expensive, massively difficult in the sense of many of these devices that are exceeding in and remote locations that have poor to no network connectivity.

I think the shift is the reason why the rebranding to edge compute is not just to clean up the notion of IoT, but to also imply that things are actually happening at the edge. It's not just producing data. I mean, IoT in a sense has been around for as you said, for a really long time. Quite honestly, that was what spawned the big data era, if you really think about it. I mean, all these things at the edge producing data caused the data volumes to balloon and eventually became out of control and then that's where you had Spark, which produce things like massive distribution, clustered distributing, data processing and things like that. Now we're shifting to the realm of let's actually do something at the edge. Let's make the edge actionable and make the edge itself smart.

[0:46:33.6] JM: I'm always curious about the business strategy around software engineering-heavy companies. TIBCO is an interesting one, because you're a very successful company. You're not exactly a major cloud provider. You're not one of the AWS, Azure, Google Cloud companies, you're also not a startup.

I've had lots of conversations with startups about how they are competing with the major cloud providers and it's always interesting, because the major cloud providers have such an advantage in terms of sales channel, because they've got all these established customers. On the other hand, they have so much surface area that they need to cover and it's really hard for them to actually figure out what is their core competency, other than – I mean, they have a lot of core competencies. They have so much surface area to cover. Where do you find the business opportunities and where do you see is the best business opportunities for TIBCO as you move into the future?

[0:47:32.5] ME: Yeah, that's a fantastic question and this is this is clearly part opinion, so I'll preface it with that. If you really historically look at TIBCO, TIBCO has fundamentally been that company behind the scenes that makes sure the major players in the industry are able to deliver on their promises. We connect and operate against the data and the systems to make sure, I don't know, your kid has their Christmas presents under the Christmas tree on Christmas Day that's powered by our technology. We've traditionally built technology that powers the world's major players.

We're not necessarily interested in competing with cloud vendors like AWS. We want to work together with AWS, because they provide something immensely valuable to the table, and we provide something that they don't, which is the technology that enables all of these other major players to facilitate their promises as I alluded to earlier.

[0:48:28.3] JM: Leon, anything you want to add?

[0:48:30.6] LS: I mean, if you look at what Matt is saying, I fundamentally agree with him. I mean, I don't think we want to compete with the likes of for example, AWS with whom we have an amazing partnership. If you look at the innovation that's coming from the joint companies, if you will, it's much more than any of those two companies would have been able to do it alone. Rather than competing with it, I would say that the offerings that both of us have would definitely complement one another.

[0:49:01.6] JM: Yeah, I'd agree with that. I mean, there's so much to be done in helping out insurance companies, telcos, banks, these companies that are making their way towards being software companies, towards being AI companies, these companies with a tremendous data advantage that need a whole lot of help in terms of revamping their technology and building software company-like processes. I see it at least, much more as a positive sum opportunity for partnerships, rather than this zero-sum competition for deals.

I mean, people might see, for example the Pentagon's competition for the 10 billion dollar cloud budget and they imagine that this is zero-sum battle between cloud gladiators. For the most part, that's not really how it works. People are just picking and choosing technologies and those technologies are working better and better with one another.

[0:49:59.1] LS: Absolutely. I agree with you a 100%.

[0:50:01.9] JM: Cool. Well guys, thank you for coming on Software Engineering Daily. It's been really great talking to you about Flogo. Is there anything else that you want to add about the project?

[0:50:08.6] LS: Yeah. I mean, flogo.io, or hit us up on Github. We've got a Gitter channel, so if anyone's interested and/or if we've confused them completely, definitely feel free to hit us up on Gitter or Github and happy to chat.

[0:50:21.1] JM: Awesome. Well, thank you Matt and Leon.

[0:50:23.4] LS: Thank you.

[0:50:24.1] ME: Thank you.

[END OF INTERVIEW]

[0:50:28.0] JM: For all of the advances in data science and machine learning over the last few years, most teams are still stuck trying to deploy their machine learning models manually. That is tedious. It's resource-intensive and it often ends in various forms of failure.

The Algorithmia AI layer deploys your models automatically in minutes, empowering data scientists and machine learning engineers to productionize their work with ease. Algorithmia's AI tooling optimizes hardware usage and GPU acceleration and works with all popular languages and frameworks.

Deploy ML models the smart way and head to algorithmia.com to get started and upload your pre-trained models. If you use the code SWE Daily, they will give you 50,000 credits free, which is pretty sweet. That's a code SWE Daily.

The expert engineers at Algorithmia are always available to help your team successfully deploy your models to production with the AI Layer tooling. You can also listen to a couple episodes I've done with the CEO of Algorithmia, if you want to hear more about their pretty sweet set of software platforms. Go to algorithmia.com and use the code SWE Daily to try it out, or of course, listen to those episodes.

[END]