EPISODE 587

[INTRODUCTION]

[0:00:00.3] JM: Gloo is a function gateway built no top of the popular open source project, Envoy. The goal of Gloo is to decouple client-facing APIs from upstream APIs. Gloo is similar to an API gateway, which is tool that software companies can use to collect all of their APIs in one place and impose security, monitoring and other standards around those APIs. The goal of Gloo is to provide all the tools necessary to glue together traditional and cloud-native applications.

Idit Levine is the CEO of Solo.io, a company that is building Gloo and several other projects. Idit was previously on the show to discuss microkernels, and it was great to have her back on the show. I hope you enjoy this episode.

[SPONSOR MESSAGE]

[0:00:57.0] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes. You can quickly provision clusters to be up and running in no time while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked into any one vendor or resource. You can continue to work with the tools that you already know, such as Helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications off-line. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes

and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes. That e-book is available at aka.ms/sedaily.

[INTERVIEW]

[0:02:32.2] JM: Idit Levine, you are the founder and CEO of Solo.io. Welcome back to Software Engineering Daily.

[0:02:37.4] IL: Thank you so much for having me.

[0:02:38.9] JM: The last time that we spoke, you were at EMC, and since then you've started your own company, Solo.io. How has the transition been from working at a large organization to running your own company?

[0:02:52.7] IL: That's actually interesting. Actually, over my life, I work in a startup company. Actually, the transition to actually EMC was really strange, because it was a big company. So I'm basically back home, because I, all my life, did startups. Yeah, I much like it because I feel that you can go much faster. There's not a lot of the overhead of a big organization. Again, there's a lot of advantage to being EMC, for instance. Much easier to get – I don't know, coverage or people to listen to you because you're coming from big organizations, so there is a tradeoff.

[0:03:26.0] JM: Is this the first you started a company or had you started a company before?

[0:03:30.1] IL: No. That's the first time I started it. I worked in companies, but I never started. That's the first one.

[0:03:34.9] JM: Right. How is the experience of starting a company differ from being an early employee at a startup?

[0:03:41.5] IL: I mean, it's much harder, specifically because I'm the only founder. You need to do all the work. So it's everything on you, so you need raise the money and you need to go to investors and you need to find space to seat. So there is a lot of work that you do need to do in

the beginning to kind of like jumpstart it. There is much more responsibility, right? You need to do a lot of stuff, but also you're responsible for this if this is [inaudible 0:04:03.4] this is on you.

The same play, you do want that to succeed a lot, but you have less skin in the game.

[0:04:10.5] JM: Right. When you started Solo did you know what you wanted to build or were you tinkering within a universe of ideas?

[0:04:19.2] IL: Actually, these markets move so fast. When we started in the beginning, what my plan was actually even before I got the money, what I pitched was regarding serverless. Actually, I had this ID for like over a year and it's moved, right? So when I kind of like came to a point that I actually got the money, I couldn't – I had to actually – It twisted a little bit. We actually extend the ID. Before it was only about serverless and then I saw the movement and I also looked, then I figured out that this market is moving on the serverless but not as fast as I wanted it to move. So I said, "We need to also to kind of fact include the current situation of the market, which is also monolithic and microservices. That's what we did. We basically took something, and that's how Gloo basically was born, right? Basically, in the beginning, Gloo was – Well, we're probably going to do it only on a serverless," and then you say, "You know what? Actually, the biggest problem in the market right now is actually to move from monolithic, to microservices, to serverless. So why if we're just going to extend it to be able to do everything?"

[0:05:23.3] JM: Right. You can see where things are going. You know eventually people are going to want to deploy applications in a serverless fashion where they're not addressing specific containers. They're just deploying their applications as collections of functions and they can perhaps choose whichever cloud provider they want to deploy these functions on to. Maybe they do it based on price. Maybe they do it based on performance or some workload specificity, but we're not there today. We're still in – We're significantly far from that future.

In fact, like you said, the space moves very fast. Not only does it move very fast, it's very hard to tell exactly where we are in the timeline. How far – Because, I mean, serverless, people talk about it a lot, but as far as – When I was on Cube Con recently, plenty of enterprises are eagerly pouring money into buying a Kubernetes provider or a collection of Kubernetes security or Kubernetes monitoring, and that's where the money is right now, but it's very clear that

eventually the enterprises will move towards more serverless stuff, but it's just not there yet. So it sounds like you kind of wanted to head your bets a little bit and get into a market that today would have some customers.

[0:06:46.4] IL: Yes, exactly. As you said, right now I'm a startup, which means that I actually also need a customer. When I was EMC, that wasn't my case. I could actually do whatever crazy project that I want, because my job wasn't to bring customer. We knew that we'll have customers. That's not a problem.

Here, I need to have customer and when you're looking at the market then you'd understand, what is the pain point right now for customer. It's actually not in serverless, right? They are still struggling with moving what they have currently, which is monolithic to microservices. But I'm a big kind of like open source person. I really like innovation and I want to help define the future.

So what I did is I said – And that's exactly where it's coming from, is, "Okay. I want to move customer to where we want to go, but I also want to play in the ecosystem of the open source, because that's where I'm usually playing." What I did is exactly how Gloo born. Basically I said, "Well, I'm going to fix the problem have right now, which is for monolithic to microservices," but I know that in the open source there's people right now have microservice already, specifically the Gloo of the world, but they might wanted to kind of like also move to serverless. Basically, glue this environment. That's why Gloo, right? Basically, glue this type of application together to one application. That's basically all use case. That way I'm playing, I can get customer definitely because I don't know any enterprise customer that doesn't suffer from this problem today, which is the movement? But I'm also bringing something to the open source community, which is interesting and it's also problem to solve.

[0:08:29.8] JM: Right. If I think about your product, Gloo, or the thrust of your set of products, the idea is that let's say you are Procter & Gamble and you have all of these legacy applications internally and you've also got perhaps some newer applications. Somebody within Procter & Gamble has spun up a Kubernetes cluster and it does new things and it does new things that would ideally leverage the old things. Perhaps you've got an old things. Perhaps you've got an old inventory management service and you've got some API in front of it that you've built, but you need to be able to glue together that legacy API and your newer Kubernetes installation, for

example. So you want to be able to glue those two things together, or maybe you've done MNA and you've got three different companies you just acquired and you want to glue those acquisitions to your existing applications. Are those some of the use cases that you're trying to appraise here?

[0:09:34.0] IL: Gloo can do that as well, but what I think that is more important is the fact that a lot of the enterprise today that has a monolithic application, everything that they're doing new, which is greenfield, they will do on Kubernetes, right? But the old things are still there, and usually what they're doing, they decided to rewrite it, refactor them? That's the biggest problem today.

Now there is two way to do that. The first way to do it is just said, "Well, we need to get a point – We're writing all these monster monolithic application to a microservices." That's one option. But that a lot of the time is just not successful, because there is a lot of functionality. It's very complex. More worry is that even if you do try to do that, you're not writing any new feature this time.

I mean, I had a customer for instance that he had a monolithic application and they really wanted to move actually all the way to serverless. So they basically went to the boss and said, "Look. We want to rewrite it. It would be so much better. We can run faster. Everything will be so much easier for us." The boss said, "Tell us how much time it will take it."

They went and did a research. They came back. They said, "Well, it will take us nine months." The boss said, "No," and he was right, because that means two things. A, it will not going to take you only nine months. It probably would take a year or a year and a half. The next problem is that on all these year, I didn't get even one feature, which means that I will get fired, right? It's like you're innovating a new room, but you're only doing piping. In the end of the day, you can see that.

That's basically what we're trying to do. What we're basically telling our customers, "We want you to extend the new functionality of this application to microservices and serverless, and we will glue it it's one application." Then a new spare time, because we're carrying all of – I will explain in the second, but we're carrying basically – We're breaking your monolithic application

to function, which is really small unit of compute. You can actually move them piece by piece gradually when you have time. You know, it's one of these. I mean, they need to learn new tools, right? I mean, if you're doing monolithic application, you will use something like [inaudible 0:11:48.0]. You will have – I don't know, Splunk for logs, and you will have probably SOA architecture.

When you move into microservice, it's all bunch of new tools, because now it's distributed application. So you will use something like Prometheus and will use something like open tracing. They need to learn, and basically what we give them is time, because we tell them, "First of all, you're already going to the right direction because we're basically adding serverless and microservices in the new features. Also, we're giving you the ability to gradually move it with no rush."

[0:12:20.6] JM: Okay. The company is Solo.io. The main product is Gloo. Explain what you have built with Gloo.

[0:12:30.6] IL: Okay. First of all I will start by telling that Gloo is a platform, and the point is that this is the base project – It's a Kubernetes in the terms of like we needed it in order to build what we really want, and now we're going to come with a lot of small product that you will see in the market.

So what is Gloo? So basically we are in the open source, and Envoy is the best proxy. I mean, maybe let me start by explaining what we really want to do, okay? When I look at all these ecosystem, the most important to me is to find the smallest unit of compute and basically try to break all the rest of the application to the smallest of compute and then I can mix and match it.

So if you're looking today, you have monolithic microservices and serverless, which is basically function, but if you think about it, the monolithic is also built from exposed API function. Actually, also the microservices is basically build from exposed API function. So if I actually manage to discover all of that in the monolithic, in the microservice and in the serverless, all the function, now I can actually assemble application from this pieces. So what I need to do in order to do something with this, we need to, A, basically break the application, which mean discover the function and be able to route them, and that what we do in that, we do things. We have a

discovery service that it's, A, discover all your infrastructure in terms of the regular upstream. So like Kubernetes and [inaudible 0:14:07.8] and everything we discover, and also your AWS instances.

Then the other thing that we're doing, we're discovering functions. So if it's – Serverless, it's easy. Just a Rest API. But if it's not, we're basically supporting Swagger. So Open API basically, and GRPC. So if you're using one of those, we'll be able to basically get the specification of the functions. Now assuming that I took all your application and managed to break them to – Not to break them, but discover the function in them. Now, the only thing that I need is to basically be able to route those function, and that's where we use Envoy.

So what does it mean route to a function? Envoy is the best for my opinion proxy that exists there. It's worked very well with our use case. The only thing is we wanted to extend it to route on the function level, because today every proxy that exists usually routes from route to an [inaudible 0:15:04.8] service. So what we did, we basically used the filters of Envoy and extend the filter with a base filter to actually route on the function level. So now every feature that you can get today in Envoy under regular level – I don't know, limiting security and so on, transformation [inaudible 0:15:26.5] and so on. Now you can actually get it on the function level. Imagine canary of function though, not services.

So we did that. We extend that. Besides that, we wanted to extend, we wanted to make it very easy for the user and we thought that if we will be on the request but we would be able to manipulate the calls. So what we did, we basically put a lot of – Created a lot of filter that relatively – It's basically working with a relative function. For instance, a call for AWS, we de-signature already. Transformation filter to transfer from any language, from JSON for instance, to GOPC, to HTTP and so on. We did all of these.

What we created is basically we took Envoy and extend its functionality to walk very well with functions and route on the function level. That's the first thing that we did. But now it's Envoy, so we basically need to manage it. While we build, we build Gloo, and what Gloo is basically it's a that watching three things. The first thing that it's watching is upstreams and function, basically the discovery that we're doing. The second thing that it's watching is the secret. So we're supporting right now Kubernetes secret and Vault.

The last thing that we are watching is the configuration. These are configurations. We're using right now either YAML, CODs or Consult, and if one of those change, Gloo getting this information, but it doesn't understand anything, and the only that it's doing is basically saying it to a plugin. Basically, all the logic of Gloo is in the plugin, which means that it's really extensible. For us, to add feature support something, the only thing we need to do is basically add a plugin and maybe a filter in Envoy and we basically define our language. We extend our language.

[SPONSOR MESSAGE]

[0:17:23.2] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[INTERVIEW CONTINUED]

[0:18:45.0] JM: From a technical point of view, I understand what you're building. I want to put it in the context of a large enterprise that is doing this cloud native or a microservices migration to their – They're doing containerization. They're building out CICD pipeline. They're instrumenting

observability across the application. Where in this process would they start to use Gloo and how would that help them?

[0:19:20.8] IL: So the first thing that they will do, they will install Gloo and it will discover all the environment. No matter how much cluster there is or which type of platform it is or where it is, which cloud it is, they will discover all the application that's running. So now if we know the application that is running, now they can start to manage them. What's good about Gloo is that you basically – That way, and I can show you a demo with it, is basically you can actually take a monolithic application – I don't know, pet store or something like that reading in Java and start basically extend by writing – I don't know, serverless or microservices and then tell Gloo, "Now, extend that."

Basically, if you think about – We have multiplexes. So we totally separate the client from the server. Basically, can take this monolithic application and say, "Now, I write an extension piece in microservices." So that you basically can tell Gloo to extend the API. Now you basically have one application that is part of, it is going to monolithic and part of it is going to microservices.

Then the next step will be – To start to rewrite. This organization will go and say, "Well, I'm going to take this piece and I will rewrite it." Again, Gloo will be the one who basically make sure that you will basically route it to a different place, to the new functionality. That's that.

Besides that, we also – In the request part, and because we're on the request part, we can help you with other things. For instance, right now you're moving your application from monolithic to microservices. There will be a point that most of your functionality will be in monolithic, but some of them will be in microservices or serverless. So the big question right now is how you monitor it, right? Why are you using the out that is going to work? You need to monitor. You need to get the logs. You need to debug it, deploy it, and it's now in three separate place.

So what we're doing is because we're under request, but we basically have the ability, because we're evoking every function, we actually suck the information from the other platform and bring into one place. For instance, if you're running application that is in monolithic running on VMWare infrastructure, then you have Kubernetes cluster and you're running one microservices. Then you have also AWS Lambda that you're running. What we're going to do,

we're basically going to suck the logs from all these places and suck the metrics and basically give it one place. It will be – I don't know. For instance, Prometheus, or DataDog or anything else. That's that.

The same thing with open tracing. We're basically going to take all the information from the function and the monolithic and the microservice, fit it to open tracing. So now you can see actually a spin that some of it monolithic, some of them in microservice and some of them in serverless, which I think really cool. The last thing is that we also build Squash, which is basically debugger for distributed application, which you can actually attach debugger. I mean, we're doing it very, very nice user interface. From the ID, you can attach the monolithic application reading in Java to a Java debugger. You can attach the microservice reading in Go to – Again, Squash is doing it, to a DLV, and then Node.js to the function and then you basically can debug it like it's a one monolithic application even though you jump and cross [inaudible 0:22:49.6]. If it makes sense. It's much easier to show it than to talk about it.

[0:22:53.6] JM: Yeah. No, I believe you, and people can definitely check it out by going to solo.io. So can you help me understand how a company would integrate with Gloo? Where along the path do I spin it up? So you said, it can discover all of the services that I'm running. How does it do that? How does it plug into my existing infrastructure?

[0:23:19.7] IL: So if, for instance, let's just take an example. I say that you have a VMWare and you have a Kubernetes and you're also using AWS, just fact of it, for the example. So we're basically going to Kubernetes, ask for the upstream. That's easy. Then we're also looking at each of the upstream to see if you have any function that we can recognize. For instance, if you're running on this service – On Kubernetes you're running, let's say, application with Swagger. So we collect that as well. We're basically going to find all your workload, cross clusters and then besides it also the function that's running on top of them and also the regular application they're running on top of it.

Now if we know that, we can act as a regular API gateway, because by far the way we build it, I think that there's no question. We are the best API gateway that exist today in the market. Also, we'll give you a lot of tools to actually immigrate those applications in case you want. Again, you

decided that you want to take application A and you want to move it to a microservices. That's where you will basically use Gloo. That's one option.

The other option is that if you look in a lot of the offer today, like for instance just like with Cloud Foundry. They're using Kubernetes and Cloud Foundry today. So they have like the container runtime and also the application runtime. There is nothing that make them work together besides [inaudible 0:24:52.6] who is deploying them. So we need something to glue that together. One place that people can actually see all the workload and maybe create an application that some of them living in this cluster and the other one, living in that, the other cluster. That's where Gloo can help as well, right?

If you think about it, there's so much offer today in the market. There is Docker. We're also giving Docker Swarm, and there is a DCOS, a mesosphere that basically offering these COIs, but also Kubernetes. There is Cloud Foundry as we said. There is a lot of platforms today that when you're building them, they're basically giving you two different cluster that are totally [inaudible 0:25:33.0], and I think where Gloo can help is basically glue them together as well.

[0:25:38.0] JM: You mentioned this term API gateway, and I think it's synonymous with function gateway. There are probably some people in the audience that don't really know what these are. Can you describe what the purpose of an API gateway is?

[0:25:51.2] IL: Yeah. API gateway is not new, right? It was like a – I think there was a lot of company that spin up before, like [inaudible 0:25:58.3]. Basically, the API is the ID of instead of managing API of each server – You have a lot of workload and each of them has an API. They basically abstract that and manage that in one place. Now you have a way to actually configure it down on, for instance, security. We can see why. You can manage them and make sure that they are secure and so on. This is the API gateway.

What we're doing, we're doing exactly the same thing. So we're allowing to use it as a regular API gateway, which means that a client come with a request. It should go through the API gateway and you can route it to the right place.

We also can offer you the same thing on the function level, which means that, first of all, you can go and call function from a serverless, which we already sign in and so on and making the call for you. Also, you can go to a Swagger function and so on and get all the function, again, that API gateway usually giving you. For instance, first of all, it's giving you metrics and monitoring and tracing. That's something that usually they're giving out of the box. But they're also giving you all of the regular functionality of the security that they're putting on top of it, the management itself, transformation sometime and so on.

[0:27:17.6] JM: The purpose of an API gateway is like I'm a company and I have got five development teams. One is building the sales software and they've got their own servers and containers perhaps, and then there's a marketing software team and they're building the marketing software. Maybe it's on Cloud Foundry, and both of these things – Both of these teams are going to expose APIs and the API gateway is the place where you can aggregate all of these different APIs and not really think about what backend infrastructures are running on, and the API gateway is a good place to standardize things like security practices.

[0:28:00.4] IL: Yeah. Security management and orchestrator, right? That's basically the three things that you will want to manage by an API gateway. Again, it's one place that the administrator said, "I have five team. A lot of application. But in the end of the day, I can see and manage all the API in one place and make sure that you are in some."

This is what this API gateway. What we're putting on top of it is the fact that we also basically, because we want them to move to one of the [inaudible 0:28:28.1] we wanted to apply is to the migration between monolithic and microservices. It was important to us to do everything that people are doing on the serverless level, but on the function level, that way there is smallest pieces that they will to migrate each time, and that's giving them the ability to do it more gradually than actually rewrite all servers, if it makes sense.

[0:28:49.5] JM: So does a typical company that would adopt to Gloo, they already have an API gateway and you're trying to not necessarily give them another API gateway. You're trying to give them a function gateway which is something that does not replace their API gateway.

[0:29:09.0] IL: So we can do both way. I mean, we actually complement the API gateway that they have, the current one, but to be fair, I do believe that Gloo is the best API gateway that exists. So I will commend them to already take that as also the API gateway and get them both functionality. Again, we also can work side next to the legacy API gateways.

[0:29:34.5] JM: Right. If I already have a legacy API gateway, to help illustrate what you can do, what would a function gateway, the Gloo function gateway, what would that be able to do to complement the existing API gateway?

[0:29:47.8] IL: So there is a few way. One way is just we're going to suck the information from the regular API gateway. Basically do a chaining of proxies, which is not the nicest way, but if you really, really want to use your API gateway, that's one approach to do, which is basically a chain of proxies, which means that every time that [inaudible 0:30:07.2] will come, it will go the one API gateway, which is the legacy one. Do the routing based on this, and then it will move to Gloo. It will basically apply it more in the function level. That's one option. The other option is just basically to totally dismiss the other one, which is basically just to suck the information that he has and basically take over.

[0:30:29.7] JM: And if I were to just replace my existing API gateway completely with Gloo, what are – And I think we touched on this a little bit earlier, but let's revisit it. What are the additional things that I will get out of the Gloo API gateway?

[0:30:46.9] IL: Again, I think the difference between us or any API gateway is the use case that we are targeting. All the thing that we're doing in Gloo today is all about migration. Take your monolithic application, transfer it to a microservices, transfer it to a serverless, right? Basically that's what we're targeting.

A lot of these features that you will get will be on that level. I will give you an example. If we already, basically, taking all the infrastructure and kind of breaking it to function, not really, but basically you can call it that way. So a lot of these stuff happening on the ecosystem of the function, like for instance, event-driven architecture. It's very popular in functions, right? We can bring it to all ecosystem, because now everything is a function.

Something like workflow that's very popular in the function level. Imagine that now you can actually chain functions, but this can also be that the function is part of a monolithic or microservices. Again, much easier to show, but yeah. You will get a lot of that functionality on top of it. Besides that, you're going to get the ability to – As I said, to extend your application in functions and the ability to actually bring also – For instance, for event that we're doing, we actually created a filter for [inaudible 0:32:05.3], which mean that basically we can that way create an event and bring it to all the other ecosystem. Like for instance, say, monolithic application or microservices can be driven by events now.

A lot of this functionality, none of the API gateway exist today. Besides that, as I said, most of the regular API gateway are not going to give you a discovery. That's not something that people are usually doing. The other one that we're giving that others is not giving is a lot of – As I said, we can do a lot of transformation, because we're doing it on the function level because we know what the function is, so we can actually transform. We're doing a lot of security, which that everybody will give you.

[0:32:48.7] JM: Sorry. Could you dive in deeper into that transformation? What do you mean by that term transformation?

[0:32:53.7] IL: I mean that you can actually do request or respond transformation. Let me give you an example. Today when you're actually running your Lambda, usually what it's giving you back, it's a JSON. But if I want this response to actually show in a browser, I want to return an HTML for instance, that's something that we will do for you. Basically you can say, "I want to transform this JSON request to an HTML," and basically – Again, be on the request path, so we can do a lot of the mediation that way. So that's an example.

Let's say that you have a backend of GRPC. You wanted your backend to a GRPC, but you're still using the old client, which is not a GRPC. So what we basically can do is transform it between that, and that way you will not need to write a full new client GRPC, if it makes sense. That's kind of like a lot of the steps.

Basically, what the transformation is giving us, which I think is really strong, is the real way to decouple the client from the server. Today it's not like this, because I will give you an example.

Let's say that I have a server, and in the server, say, you know the domain and then I have the API [inaudible 0:33:57.1]. Basically, the way it's worked with most of the proxy, they will remove the initial domain and so on, but that mean that on the backend of the service, you need to have these basically pat. So if you have [inaudible 0:34:12.0] version 1, blah-blah-blah-blah, you need to have this part on the service. Not in our case, because we totally decouple it. That's another example. Basically we'll decoupling all the client form the backend. That's giving you a lot of use cases for transforms, again, transform JSON, [inaudible 0:34:34.6] and so on.

[0:34:35.6] JM: Again, the API gateway works off of Envoy, which is a service proxy. We've done a show about Envoy, if people want to check that out. Explain how the API gateway utilizes Envoy.

[0:34:48.3] IL: Basically, Envoy is our proxy, which means that we're basically defining a [inaudible 0:34:53.2] request. What is the route that it need to go and we're putting some filters on it. Basically, we're getting the request and we can route to the right place, but we can also do a lot of – Fly a lot of stuff on it. For instance, we can – As I said, influences in AWS. We're basically putting the signature because we know the secret. We're basically putting the signature so that way the user doesn't need to take care and all about applying keys and so on.

We could do a lot of other stuff. Basically, what we're doing is writing filter. This is the way to work with envoy, and you can put a lot of functionality, and functionality that will imply on the request part, on the [inaudible 0:35:34.3] actually as well.

So that's what we did. This is basically the proxy, but now you need to manage the proxy. You need to tell them where to route, how to route, and that's what basically Gloo doing, all the API gateway, is basically is giving a configuration to Envoy to know what to do basically, if it makes sense. Also, because it's Envoy, so Envoy already have filter like statsd and open tracing and so on that we can actually leverage metrics and so on. Again, we will configure that. What Gloo will do is we'll configure it, but in the end of the way Envoy itself is the one that will get the request and act on it.

[0:36:16.2] JM: How does Gloo compare to the service mesh model like Istio?

[0:36:23.4] IL: Okay. So Istio is a service mesh. That means that on every microservices that's running on your infrastructure, on your Kubernetes, you're also putting next to it a sidecar with a proxy. Then you also trick this sidecar to always talk only to this Envoy, which means that way all the traffic in and out this service is going through Envoy. So it can manipulate now.

Now we mentioned that you have actually this all over the cluster, now you basically can create mesh, because all those envoys' sidecar capable of talking between themselves. That's what service mesh is doing. What Gloo, right now, is basically using it as an ingress. By using an ingress, it's doing some stuff that maybe you need to do in a mesh. That's number one. Number two, it's not deploy as a mesh. We mean, when we are routing for Envoy, there is one Envoy that routes you to this service itself, to the microservices. But to be fair, in Gloo we can actually also do service mesh. I mean, it is a service mesh. We're just not using it as one. Basically we can give a multi-configuration to different Envoys. For instance – Yeah, we can be a service mesh. There's also Istio. So there's no point.

[0:37:44.5] JM: Can we take another top down example? Can you explain how a larger organization would start to use Gloo? Or maybe if you have an example of somebody who have adopted it or is experimenting with it, how they have integrated Gloo and how they've made use of it.

[0:38:01.8] IL: Yeah. As I said, I mean, a good example is a customer that basically wanted to move from [inaudible 0:38:06.7], because as I said this is the target that we are basically targeting. So that would be our customer. They basically wanted to move from monolithic to microservices. They actually decided to use as the API gateway. That was kind of like easy for us. Basically put Gloo, we discover all the environment. They wanted to move – Or actually they want AWS, and what they wanted to do is to migrate their monolithic application to Kubernetes. What we did is we basically went in Kubernetes, discover also the instances of the AWS. What we did, we basically managed them both. So all the API, all the migration and so on. Then we basically let them extend new functionality on Kubernetes. When they wanted, they would be able to rewrite the pieces and basically move them to a microservices and all of these had been managed by Gloo in the terms of like you can apply – You're basically changing the route and it's working. Envoy is so good that it's like we're using the ADS [inaudible 0:39:08.1] API. So it's so fast. It's so efficient. So you can easily change the – Don't to go these microservice – Those

monolithic. Just go to this microservices. That's an example of basically nicely migrated application for monolithic to microservers.

If you think about it, this is – The reason that Envoy was built, right – I mean, every company that did a real migration right this, from Twitter, to Lyft, basically they were using proxies and I think that the reason Lyft was actually building Envoy [inaudible 0:39:39.3] client, build Envoy is because of exactly this. They wanted to move from monolithic to microservices.

Today everybody's talking, an Envoy is to use case. Usually using service mesh context, which is visibility, observability and security. But I think that they are missing the fact that the first [inaudible 0:40:00.2] or actually build it was migration, and that's where we kind of like leveraging.

[SPONSOR MESSAGE]

[0:40:12.4] JM: Every team has its own software, and every team has specific questions about that internal software. Stack Overflow for Teams is a private secure home for your team's questions and answers. No more digging through stale wikis and lost emails. Give your team back the time it needs to build better products. Your engineering team already knows and loves Stack Overflow. They don't need another tool that they won't use. Get everything that 50 million people already love about Stack Overflow in a private security environment with Stack Overflow for Teams. Try it today with your first 14 days free. Go to s.tk/daily. Stack Overflow for Teams gives your team the answers they need to be productive with the same interface that Stack Overflow users are familiar with. Go to s.tk/daily to try it today with your first 14 days free. Thank you Stack Overflow for Teams.

[INTERVIEW CONTINUED]

[0:41:25.6] JM: Right. Okay. I think I understand it now. So Envoy is a service and you use Envoy, you instrument all of your different services with an Envoy agent and what Envoy allows you to do is things like changing your routing on-the-fly or doing circuit breaking or being able to – I think, with Envoy, you can do things like routing, a percentage of your traffic from 30% of your traffic to the new service so that you can – If you're rolling a new instance of a service then

you want to route 10% of your traffic to it to make sure it's stable before you route 100% of your traffic to that new service.

So you've got all of your containers or your servers instrumented with Envoy, and what you're saying is that Gloo is useful for a migration because you want to maintain all of your API – From your API gateway, you want to have all of those have constant uptime while under the hood Envoy is changing the percentage of routing traffic. If you wanted to migrate from your monolithic application to breaking that up into microservices, then you got to want do this gradual percentages of traffic changes in order to test it overtime and gradually move it to larger and larger percentages to the whatever medium you're migrating to.

If you're migrating from a monolith to microservices or you're migrating from an on-prem system to the cloud or doing hybrid stuff or you want to route the – And you have bursty workload, maybe you route the burstiness to the cloud, for example. Am I understanding things correctly?

[0:43:16.9] IL: I mean, that's one of these case, but [inaudible 0:43:18.6] Envoy a lot of others like retry, circuit breaking [inaudible 0:43:23.8] traffic shifting, blue/green deployment, canary deployment, all of these you will get. But we're also doing that more because we want also the ability to actually change the route, right? Again, on the function level, to the – When you're actually immigrating. Right now, if we're taking it and we're saying we have a monolithic application. There is 10 API for it, okay? Just [inaudible 0:43:46.5]. And I want only to change the functionality of one of them and we will be the one that will say every time that someone bring me a request, I will go to the monolithic application, but this function that when I will get request, I will go the microservices.

Basically, you have one application that some of it is the monolithic, some of it is the microservices and so on. That's giving you just – Give you time to actually, instead of taking and do a blowout kind of like rewrite of all the monolithic, you can actually cut it to an API calls, if it makes sense, and take part of the application. As I said, this is the migration that's being done everywhere in Lyft, in Twitter and some.

[0:44:27.9] JM: Can you describe that a little bit more? I think I'm a little bit confused. I didn't understand your last point.

[0:44:33.3] IL: Okay. Let's say that right now I have a Java monolithic application. Okay? It expose 10 function. Okay? 10 function that you can call. I don't know – Let's make it easier. Let's say that I have [inaudible 0:44:43.8] of application and it's doing plus, minus, multiply and divided, and so I have four function that actually expose under monolithic application.

Now let's say that I wanted to migrate this application. One way to do that is just to go and rewrite all of these as a microservices. This is a simple example, but they already relate to some of these. I need to create microservices for it – You know what? Four function. Let's go, because it would be easier in the serverless.

Okay. That will take time. Usually it's not that simple application, which means that when it rewrites all these applications, it will take me time. That's what people are doing today. They're trying to take these monolithic application and rewrite all in either function or serverless. What Envoy is letting you do is basically say, "Okay. Let's continue route. Every time that's someone doing plus, minus or multiply, still rounding to the monolithic application." But then I will take the divide function and only it I'm going to rewrite it and put is as function, like a serverless function. Okay?

What will happen right now is this; I have this monolithic application. It's still theoretically – You're still function of it, but only when someone will call plus, minus and multiply, it will go to the monolithic an when the customer will – The client right now will call divide, it will go to the function. Basically, running one application to the user, it's the same thing, right? You see [inaudible 0:46:19.7] slash something. It totally looks to him like a regular client, but on the backend what's happening is that three of the function will be called in the monolithic application. One of them will be called on the serverless function.

Now, for these – Now you have this application which is totally separate, right? It's different type and so on. You still want to retry. You still need to do circuit breaking. You still want to make sure that right limiting is working. You still need to do the traffic shifting, blue/green canary. All of these, you still want to be applied. It will, but now it will do it – Even though that your application is spitted around, it will – Basically, Envoy will take care of both. I hope it makes sense.

[0:47:00.6] JM: No.

[0:47:01.9] IL: It's much easier to show them.

[0:47:05.7] JM: Yeah. No, I understood. I want to talk about the broader Kubernetes ecosystem, because you have an idea for how things are going to go from the current popular model of migrating towards Kubernetes to more people adopting serverless and using that more aggressively, and I think one point of debate that's going on right now is the whole idea around should there be a standard around what an event is, because many of these serverless functions get triggered on events.

The way that people want to implement a serverless function is an event happens. You upload an image to S3 and that event triggers a function call, a serverless function call. One of the things that people want – So some people want is an open specification around those kinds of events. For example, if I upload an image to S3, S3 would create an event that would be less S3 specific, so that if I wanted to instead use Google blog store or whatever it's called, or if I wanted to use Ceph – I think Ceph is an object store, then maybe I could have cloud events that would be emitted from that object storage that would be not S3 specific. Explain why that's important. Why is the cloud event specification important?

[0:48:41.7] IL: So if you will actually have a language or a protocol to the [inaudible 0:48:48.8] of those events, now basically what you can do is to basically treat all those cloud, like one big virtual cloud. So all the services, and basically you can orchestrate through them like a multicloud solution. For instance, imagine that I'm – I don't know, I wanted to actually order an Uber, okay? I'm clicking on this button. That will actually go to my infrastructure on-prem, do some stuff. Okay? I don't know – Look that my visa is good, then go to a different cloud, like for instance AWS, because that's where my storage is. I want to get some information. Then take this information and actually go to Google and – I don't know, use auto services that Google is supplying because it's maybe better than what AWS supply and then – I don't know, go to a zoo and do something else.

This ability to actually leveraging all the services in all the cloud will be much easier. There will be one language that all those services are going to work. Then you imagine that you will have

one big cloud and you can basically mix and match services. Okay? Take the best service that work for you, and that's also will, in a way, that you're not really locking to the cloud. Let's say that all the data is in AWS, S3, it means that the only that I actually need to run there is a one small function only when I need the data. I don't need to run my instance to there, for instance. That will give you the ability to actually not be locked provider.

Now, the problem with that is that the question if the provider is okay with that, but that's a different question, because most of these movement, it's actually been done for people who has a [inaudible 0:50:26.8] offer that is basically kind of like a project like – The guys from serverless or the guys from [inaudible 0:50:35.9] or Project FN and so on. Those people usually can run on-prem. Of course, we would love that that will happen. I'm just not sure if this is something that the cloud provider themselves would be a big fan.

[0:50:51.0] JM: Right. This is the same question I'm unsure of; would AWS Lambda specifically, would they be wanting to be open to this cloud event specification. All due credit to AWS Lambda. They invented functions as a service. They were first ones to –

[0:51:10.4] IL: [inaudible 0:51:10.4], right? The question is why should they? I mean, [inaudible 0:51:13.9] today is serverless, most likely they're running on AWS. For this specification, basically mean that people will not have to. They will be able to run Google function, but use the – I don't know why they should, because they [inaudible 0:51:28.3] this market anyway, but this politics, right? It's not related to technology.

I mean, this is a good question. For what I see – Again, I'm not like – What I see from far. I think that it makes sense that Google will do that, maybe [inaudible 0:51:44.3], because I said, they're not owning this market, but I didn't see that AWS is very corporative on this. Maybe I'm wrong. Again, I don't have all the data, but that's what I see from far.

[0:51:55.9] JM: Yeah. We'll see.

[0:51:58.4] IL: But I think – As I said, what's cool about is that basically the vision is, and the vision is right. The vision, I'm totally supportive. As I said, I mean, when we've been Gloo, that's what we – When I pitch it to my investor, this is where we started. It's basically the ability to take

all the cloud, create one virtual cloud that you can ran workflow between them, that you can mix and match and so on. The problem is that the way I see that is that the future will be so far. I mean, it will maybe happen, but it will take a long time.

So it's really cool, but I think that it will take a long time. That's why I extended it and said, "You know what? Maybe that's also what Gloo doing." Gloo is – As I said, because we build as an event gateway. We build as the ability to get events and route to different providers. So you can do that as well with Gloo, but I will argue that that will be probably more popular in the open source community than actually an enterprise running this one. I mean, I have a customer and he said to me, "We are totally ready for a lot of cloud, but not the multicloud part," which is basically going between.

[0:53:02.8] JM: So what's the go-to-market strategy? I was at Cube Con recently. There were lots of enterprises that were looking to spend money and there are also lots of vendors that were pitching to the enterprises. So you do have this two-sided marketplace where people are ready to buy and people are ready to sell. What's your specific strategy?

[0:53:27.7] IL: Specifically, for me, we open source the platform, because that's what we believe, which is Gloo. We're building stuff on [inaudible 0:53:34.7]. You will see a lot of stuff coming from us soon that are basically more specific offer. For instance, we just talk about two use case with Gloo, right? One of them is the hybrid app, which is basically – I call it hybrid app, is the ability to basically glue application from different workload.

The other one is, for instance, [inaudible 0:53:51.4] which is the ability to only do this in serverless and basically be able to get in events cross cloud and so on. That's a different offer for my opinion for different people. I don't think that enterprise will care about event gateway in that stage. That's what we're going to do. We're basically going to rock. Of course, we're going to put a lot of magic on top of this, that we're now going to open source.

But basically, hybrid app is one solution. It would be one offer for the customers. Event gateway most likely will open source it, because I don't believe that it's now – No one will pay for that. There is a lot of more that we're going to bring soon, but again, stay tuned.

[0:54:29.4] JM: Okay.

[0:54:30.4] IL: It's going to very cool, I think, that – In like less than a month, we will open source some data. For opinion, it will be huge, but we will see.

[0:54:39.1] JM: Okay. Well, I'm excited to see it. Idit Levine, thanks for coming on Software Engineering Daily.

[0:54:43.1] IL: Awesome. Thank you so much.

[END OF INTERVIEW]

[0:54:48.0] JM: If you are building a product for software engineers or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an email, jeff@softwareengineeringdaily.com if you're interested. With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers.

I know that the listeners of Software Engineering Daily are great engineers because I talk to them all the time. I hear from CTOs, CEOs, directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves. To find out more about sponsoring the show, you can send me an email or tell your marketing director to send me an email, jeff@softwareengineeringdaily.com.

If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company. Send me an email at jeff@softwareengineeringdaily.com. Thank you.

[END]