EPISODE 955

[INTRODUCTION]

[00:00:00] JM: Serverless is an execution model where applications are scheduled and deployed to servers that are not directly managed by the application developer. In serverless execution, an application only loads and operates when a user actually needs to get a response from that application. This saves on resources, because many applications do not need to run at all times. They only need to be available for user requests.

The serverless model was popularized by Amazon Web Services Lambda. When Lambda first launched in 2015, it was an experimental product. Today, it is a widely used product and the market has validated the desire for serverless execution. Other cloud providers have introduced different models of serverless functionality, including Google Cloud Functions, Azure Functions and Fastly Edge Computing.

Zack Bloom is director of product for product strategy at Cloudflare, and he joins the show to discuss Cloudflare's model for serverless execution. Zack also discusses Cloudflare's growing product line, including the fast privacy protecting DNS resolver 1.1.1.1. Zack is a rare mix of engineering, business strategy and product vision and that made for a great conversation. Zach's been on the show before and we've also done episodes about the Cloudflare serverless technology and some other various episodes about Cloudflare. It's a pretty interesting company, and you can look at those episodes in the show notes or search for our back catalog.

Upcoming events that Erika and I will be at for Software Engineering Daily. We're going to be at KubCon con San Diego 2019 and AWS Reinvent in Las Vegas. We're planning a meet up at Reinvent on Tuesday, December 3rd. That's the planned date at least, and there is a link to the meet up in the show notes. You can sign up if you're going to be in Las Vegas, and we're actually looking for a venue. If you have some space in Las Vegas at the event, then that might be useful for us, or if you want to sponsor like a big restaurant or something, that could be cool. Send us an email, jeff@softwareengineeringdaily.com. Thank you.

[SPONSOR MESSAGE]

[00:02:34] JM: This podcast is brought to you by PagerDuty. You've probably heard of PagerDuty. Teams trust PagerDuty to help them deliver high-quality digital experiences to their customers. With PagerDuty, teams spend less time reacting to incidents and more time building software. Over 12,000 businesses rely on PagerDuty to identify issues and opportunities in real-time and bring together the right people to fix problems faster and prevent those problems from happening again.

PagerDuty helps your company's digital operations are run more smoothly. PagerDuty helps you intelligently pinpoint issues like outages as well as capitalize on opportunities empowering teams to take the right real-time action. To see how companies like GE, Vodafone, Box and American Eagle rely on PagerDuty to continuously improve their digital operations, visit pagerduty.com.

I'm really happy to have Pager Duty as a sponsor. I first heard about them on a podcast probably more than five years ago. So it's quite satisfying to have them on Software Engineering Daily as a sponsor. I've been hearing about their product for many years, and I hope you check it out pagerduty.com.

[INTERVIEW]

[00:04:01] JM: Zack Bloom, welcome back to Software Engineering Daily.

[00:04:03] ZB: Thank you so much. Thank you for having me.

[00:04:05] JM: We’re going to talk about a variety of Cloudflare things today. I want to start by talking about serverless, because you wrote a post about Cloudflare Isolate. It’s actually something that multiple people have written in about because they're curious, because we’ve done a bunch of different shows about serverless related things and Cloudflare has a different approach to serverless.

Let's start with the brief review of the typical definition of serverless, which was pioneered by Amazon Lambda. Can you describe how the AWS Lambda execution system works?

[00:04:46] ZB: Absolutely. AWS Lambda is what I would consider to be a pretty traditional way of running code, which is you get a process that's secured from all the other processes running on a machine that's just for you, that runs your code. Maybe it runs node or it runs Python. What Amazon Lambda figured out though is you shouldn't have to administer or manage the machine that's going to run that code. So you give Amazon your code and they handle finding the machine to run it, making sure there are enough of them running in any given moment. Scaling them up, scaling them down. Doing all of the administration around I need a bunch of processes running on a bunch of machines that are going to serve requests.

[00:05:24] JM: A common problem with AWS lambda is the cold start problem. Can you explain what the cold start problem is?

[00:05:33] ZB: Absolutely. In talking about what Lambda is, it's the same technology we had, which is a process running on a machine, maybe running something like node, but with some better developer ergonomics. In some ways, easier ways of scaling, easier ways of paying for it, because you don't have to pay for instances anymore. You just pay for the number of these Lambdas that are running in any given moment.

The problem is the underlying tools, things like node, were never really designed to run in this kind of environment. Node in the ear that it was built, you might deploy your software once a week. If node takes 500 milliseconds to start or it takes three seconds to start, that's not a big deal, because you're going to be doing one of these deploys maybe once a week, maybe once a day if you move pretty quickly as an engineering shop.

Moving it into a world where you have to start one of these node instances in response to a request. So a user loads your app or loads your website and that request flows into AWS into an API gateway and is now supposed to start a Lambda. All the sudden, node has to start in 5 milliseconds, not 500, and it wasn't designed for that. So it really doesn't do a great job of starting that quickly.

It's a fundamental flaw of trying to map these process-oriented tools that were meant to deploy pretty slowly into a world where you want this massive amount of multi-tenancy. So many

different copies running on a single machine starting up almost instantly in response to a request, and that creates delays for users of several hundred milliseconds or a request will just take longer than anticipated. That doesn't just happen every time you deploy code. It happens every time they have to scale the number of Lambdas that are running.

Since only one Lambda can process a single request at a time, if you get 10 requests per second, then you're going to get 10 cold starts. Then if you get 100 requests at the same time, you're going to get another 90 in order to scale up all of the instances in response to those requests coming in.

[00:07:29] JM: You work at Cloudflare, and Cloudflare has a fairly different model for serverless. Describe how Cloudflare thinks about the serverless paradigm.

[00:07:40] ZB: The interesting thing is we didn't originally set out to build the serverless computing platform. The kind of origin story for me of workers, which is the thing that now very much is one is talking to – My very first customer call at Cloudflare three years ago was with an A-B testing company, and they said right now we add A-B testing to websites for people and it adds a bunch of latency.

It slows down these websites, and that's a big problem, because A-B testing is all about optimization. We can't slow down their website. Is there some way that we can take this giant global network that you have, which is what Cloudflare really is. We have servers, thousands and thousands of servers in 194 different places around the world. Very close to basically everyone who uses the Internet. They said can we somehow put our A-B testing code to run on your servers so it wouldn't delay these customers?

We told them no, and we told a lot of customers no in response to that question. Because when you looked at the economics of request, a request to Cloudflare for a given website can go to any of thousands of servers around the world. If we were going to run the equivalent of that node Lambda on all of the servers that could receive a single customer's website, we're going to run thousands of copies of it. An empty no Lambda consumes 35 MB of memory. So you're talking about something like 200 GB of memory being consumed to run nothing, to run an empty example, and that's if you only have to run one copy of it on each of these machines.

Then when you realize, if you run 100 processes on a machine, it works pretty well. But if you run a thousand processes in a machine and Cloudlare has 20 million customers, 20 million different websites rather that use Cloudflare, if you run a thousand processes in a machine, all the sudden you spend a lot of time context switching from one process to another to serve requests. If you do 10,000 and 1 Cloudflare server could easily support that number of websites, you spend all of your time context switching. The processor isn't actually running any customer's code. It's just switching from one bit of code to another.

You run into these really fundamental limitations. Cold starts is another one, where all of a sudden a customer is are paying you a bunch of money to make their website faster, maybe to make A-B testing work way faster. You have to explain that some of your customers are going to get 300 millisecond delays in loading that endpoint.

We told them no, that we couldn't do it. But eventually what very smart people figured out, and one of them is Kenton, who's been on this podcast before, is there is a technology that we could adapt into our world. The technology from the web browser, V8 JavaScript and WebAssembly Engine that runs inside Google Chrome, because the web browser is actually solving the exact same problem.

It needs to run code securely in a really, really isolated sandbox. So PayPal.com can't steal from google.com. It needs to start code incredibly quickly, because when you open a tab, you really care that it loads fast, and it needs to uses super minimal amount of memory, because you get very angry when your web browser uses up too much memory. One of the biggest complaints about web browsers is you have more and more tabs open, they get slower.

The way they solved this problem is with a concept called Isolate, where instead of having a different process for each bit of code you want to run and relying on the operating system to use the same mechanisms it's had for the last 40 years to manage who gets to run when, what if we put it all in one process? We have one virtual machine and everyone just gets a little tiny thing called an Isolate, which just represents their code and essentially their variables.

That single virtual machine can do a really smart job of balancing who gets to execute which Isolate is executing when all in a single process. The operating system never context switches. It keeps a single process running the whole time and it can spin up a new one of these little Isolates, because it doesn't have to start a new node process, a new virtual machine, anything like that, in like 4.4 milliseconds I think is the average. Something like that.

That's 100 times faster, 50 to 100 times faster than node, than Lambda, any of these existing things that a lot of people use, and it uses about a tenth as much memory, about 3 MB instead of 35. It ended up fixing a lot of these really fundamental problems, and we used it to let people configure Cloudflare and to let people start to build applications, but it wasn't until we started benchmarking it that we realized there was something here that kind of transcended what we thought we were doing, because we set up this first benchmarks and we saw, “Yeah, we’re a lot faster than Lambda. We’re faster than Lambda@Edge. That’s great. We have a denser network. We’re closer to the testing location. So that makes sense.

But then you look at the testing data from close to where the land is running. You deploy a Lambda to US-East-1 and then you run a Cloudflare worker there and you run a Lambda@Edge there and you test it from really close to there. Network latency has nothing to do with it. Everything is close, and you realize the worker is also way faster, and that was shocking, and we suddenly realized that there might be something to this idea Isolates that transcends our particular needs to run it around the world.

It might actually be a really good way to do serverless in general, because it fixes a lot of these fundamental problems like cold starts and makes it really possible for someone to just write code and deploy it and not have to think about how to get it running or running all around the world, that maybe it was time for a new type of multi-tenancy-based architecture that doesn't rely on the operating system that was stuffed that was figure out four years ago but is actually based on something more modern.

[00:13:08] JM: The contrast between the two models, between the AWS Lambda model and the Cloudflare worker, Cloudflare Isolates model, is in the Lambda world. The host spends up probably a VM and then a container within a VM or maybe just a container.

[00:13:27] ZB: It's just a container.

[00:13:28] JM: Just a container on bare metal. Okay. Then has to execute the code and has a whole stack devoted to managing that container. whereas in the Isolates model, there's just a context and there's perhaps single process managing all those different contexts on a V8 instance, on a Cloudflare box in the work or the Isolate model. What kinds of workloads can the worker model, you're the Cloudflare worker model. What kinds of workloads is that well-suited to and what kinds of workloads is it not well-suited to?

[00:14:07] ZB: That's a really good question, because both of the sides have answers. One of the interesting things is our world is not built around how processes work. We don't let you do system calls. We don't try to pretend to be giving you a process in the way that a VM does. We actually do everything with the web platform. If you want to fetch something over the Internet, you use the fetch API. The same thing you use in a browser.

If you want to compute an HMAC, you use the web crypto API. So that means that our world is much more similar to writing code in a web browser, and that's the type of code that we expect then how you would write code to run on a traditional system with system calls and opening files and things like that. Already, we're kind of thinking about a different world of code and it requires a little bit of adaptation to take a piece of code that was originally meant to run on a server somewhere and suddenly run it inside a worker.

The other big world is how much access to storage you have, because we are building out our capability to let you store data around the world and have databases, have things like that, so you can truly build an application that's deployed basically how the Internet is deployed where it just kind of runs everywhere and you have no idea. You don't need to worry about where your data is stored and where your code is running.

But right now we kind of have the code running part figured out, and the data storage part, we're just inching our way closer to. If you have an application right now that needs a ton of data storage, you have to use an external system like Firebase a lot of the time. We have a key-value store, but it's not necessarily what you're going to use for the primary data store of your application. That's one answer. If you're doing a ton of crud-type operations, you need a

traditional database. You need transactions. You need those types of things. You at least have to store your data externally and then process it and interact with your users inside the worker.

Conversely, something that is somewhat stateless works amazing in a worker right now. If you want to compute, do session authentication with an HMAC, which is one thing that I mentioned. A user comes in and they give you a token and you decide if that token is valid and whether you should forward that request along to your API. That's a beautiful thing to do in a worker.

If you need to serve content to people and you want your site to be really, really fast or you want to deploy your static website, that's a beautiful thing to do in a worker, and it's really great because you have a much nicer like growing path with a worker than you do with a traditional I'm going to deploy a static website.

If you decide you want to run some code and you want things to work a little bit differently, you can just write that code and run it in your work. It will execute it all around the world. Whereas with a traditional static website, it just kind of works the way it works and you can't really grow. So things that are at least somewhat stateless or where you can really encode the data into a key-value type model and where you care about the performance are really the best places to use a worker right now, but ask me again in a year, and it will have grown a lot.

[00:16:57] JM: I think the other interesting aspect about it is there's constraints around the languages you can use, because it if it's executing in V8, then the only things that can execute in V8 are JavaScript or WebAssembly situations where WebAssembly actually works. It's like Rust and what else is in –

[00:17:16] ZB: C, C++, like traditional compiled languages. This is one of these things that we were just incredibly lucky with, because at the time that we started doing this, totally coincidentally, the WebAssembly craze had started and all these languages were doing a bunch of work to port themselves to WebAssembly. Obviously, that's nothing we did, but it was very, very lucky for us because this V8 engine all of a sudden can't just run JavaScript in languages that compile the JavaScript, like Typescript, which we use a lot.

We could use Rust and C and C++, and now increasingly Go. Support for Go has gotten significantly better. Basically, any language you can name is currently at some point in the process of being ported to WebAssembly. I know there's a COBAL part. There's ALGOL. I'm sure there's FORTRAN somewhere, but obviously also a C# PHP languages that more people care about. That was, again, just total dumb luck, but it means that this platform isn't just relevant for people writing JavaScript. It's relevant for almost any language as long as you can compile, you have the source code and you're capable of compiling it into this new WebAssembly thing. It's not just a compiled binary that you have somewhere.

[00:18:22] JM: I keep seeing anecdotal evidence that this future of more and more stuff moving to the edge is coming to fruition, and it will be interesting to see what the convergence is or what the – If you can run anything, essentially, at the CDN layer, or at the backend server layer, it would be very interesting to see what the division of labor between those two areas is. Do you have any ideas or predictions?

[00:18:56] ZB: Traditionally – And I agree with you, that I think this is a really interesting thing. Traditionally, people thought of edge computing as a really esoteric thing. If you read some early reports on edge computing, it was all about it's going to be autonomous vehicles, augmented reality. These future things, flying cars, basically, is going to be what is edge computing for. Obviously, everyone's going to keep doing the things we already do in the core.

But with the stuff we're looking at with Isolates, all the sudden it kind of gets cheaper to do edge computing. It's not really expensive anymore in the way that it would be if you're trying to do Kubernetes at the edge, but it's way cheaper. Because if a request comes in Ulaanbaatar, Mongolia, it's actually cheaper for us to spin up and isolate in a few milliseconds of CPU, respond to the request and be done than it is to transport, pay for the bandwidth to get the request all the way to US-East-1 and then respond. So all the sudden, this edge computing thing isn't really, really expensive. It's actually often significantly less expensive, for us at least, than processing all the requests in one location.

If it's faster for users and it's less expensive and it's so much more resilient and failure-tolerant, because it's almost impossible to have 194 data centers, or maybe literally impossible, go down in the way that you could with a single availability zone or region. Any network partition or

anything like that can block users from being able to access a traditional data center. But it's totally impossible that it could happen all around the world. Why would you put something centrally? What would be the advantage of doing that centralized compute? Maybe if you needed to do some sort of really specific data processing where you needed a huge amount of data all in one place that could interact with each other.

But in our experience, what we're looking at is that a lot of applications really only need to access a subset of data. When you and I edit a Google Doc, we don't need every single Google Doc on earth to be in a single server somewhere. We just need the one we're editing to be in a location close to us. We believe that there are a lot more opportunities to divide data up into different regions where it's necessary or valuable or you want to store it for legal reasons and not need to try to have a single view of truth in a single location where all of your processing is going to happen, and that it's better for users and faster and more resilient. I think most things are going to happen and what we now call the edge. But I think in the future we'll just call like the Internet.

[00:21:19] JM: Maybe you keep your entire Cassandra or MySQL database in a backend place. Maybe you keep it close to the TP use and the GP use that are all on some cloud provider that has all that horsepower and you want the data close to all the horsepower. But for most workloads, the edge will do just fine.

[00:21:48] ZB: And it will be faster for users and easier for people to deploy in. Yeah. It will be the final evolution of moving from a model of mainframes located in the basement of an office building to the Internet to something that works the way the Internet works. Your compute and you storage should be as geographically diverse as your users.

[SPONSOR MESSAGE]

[00:22:15] JM: Being on-call is hard, but having the right tools for the job can make it easier. When you wake up in the middle of the night to troubleshoot the database, you should be able to have the database monitoring information right in front of you. When you're out to dinner and your phone buzzes because your entire application is down, you should be able to easily find

out who pushed code most recently so that you can contact them and find out how to troubleshoot the issue.

VictorOps is a collaborative incident response tool. VictorOps brings your monitoring data and your collaboration tools into one place so that you can fix issues more quickly and reduce the pain of on-call. Go to victorops.com/sedaily and get a free t-shirt when you try out VictorOps. It's not just any t-shirt. It's an on-call shirt. When you're on-call, your tool should make the experience as good as possible, and these tools include a comfortable t-shirt. If you visit victorops.com/sedaily and try out VictorOps, you can get that comfortable t-shirt.

VictorOps integrates with all of your services; Slack, Splunk, CloudWatch, DataDog, New Relic, and overtime, VictorOps improves and delivers more value to you through machine learning. If you want to hear about VictorOps works, you can listen to our episode with Chris Riley. VictorOps is a collaborative incident response tool, and you could learn more about it as well as get a free t-shirt when you check it out at victorops.com/sedaily.

Thanks for listening and thanks to VictorOps for being a sponsor.

[INTERVIEW CONTINUED]

[00:24:05] JM: Do you think we'll ever see a time where backends will get compiled or bundled and actually deployed to user applications? Imagine like I have my smartphone right now. If we can deploy a subsets of backend logic to the edge and more and more of those subsets are going to get pushed to the edge, why not push it even further? Why not get a WebAssembly bundle on my phone so that maybe I'm just hitting my client device for a lot of the backend logic? Is that realistic?

[00:24:50] ZB: I think it's realistic. You have to figure out why you're doing it. One advantage would be it's easier for developers. I remember when node was coming out and people were really excited about the idea of isomorphic development, that you could write a line of code and it could run in node or in the browser. That didn't really work out, because it turns out node is a very different environment in the browser and it has all of its own APIs, but it could work in something like Isolates, because it is – They're both literally the web platforms.

So, they support the same APIs, which is closer. But you have to be sure you know what the purpose of doing that for is. Is it efficiency or is it to have a single development environment that you get to deploy code everywhere with? Is it performance? I think it's totally something that you could do, and I do know people who are compiling WebAssembly into the browser in order to augment it and add capabilities.

[00:25:40] JM: Yeah. Did you have to do anything to modify V8 or to get V8 to do what you wanted it to do as this multitenant backend a service edge computing platform?

[00:26:00] ZB: There are a lot of pieces that Cloudflare built. You have V8, the core, which you can give pieces of code and it can run them in an isolated and somewhat cooperative way. But then you have to decide what you're going to run when. So you need a whole process runner that's going to manage these V8 Isolates and spin them up and spin them down and respond to requests. You need all of the APIs.

When you make a request with the fetch API, it's our C++ code that's actually executing. We have access to the cache API that lets you store things in a Cloudflare cache around the world. That's all C++ code that we had the write. Web Crypto is all bindings on top of crypto libraries that we had to write. It literally can just run code. It can't do anything interesting without a layer on top of it that we had to write. There's also tons of security stuff that I am not the best person to talk about, but I know was done in order to make it a secure platform as well.

[00:27:01] JM: The workers and the Isolates, this is not the only thing you work on. Your title is now VP of product.

[00:27:09] ZB: Director of product.

[00:27:10] JM: Director of product.

[00:27:10] ZB: Yeah. One of the directors or product at Cloudflare, and I focus on product strategy. So that's new stuff that we build. Particularly new stuff that appeals to totally new markets, totally new areas, customers who maybe wouldn't have been customers of Cloudflare

before and can be now, and we get to do that because we have this really big global network with all these servers running in all these places around the world. So every day we kind of show up to work and say, “Given that you have that, what is it that you're suddenly able to build?”

[00:27:39] JM: It’s funny seeing people take the perspective that the cloud game is over. AWS has won. It’s so deeply incorrect. There's so much white space remaining to explore, particularly for companies that don't have a gigantic catalog of products that they need to maintain already. Cloudflare has a smaller set of things that it has focused on and done extremely well and built core competencies in. What are those core competencies of Cloudflare that you see as the backbone of opportunities to explore?

[00:28:20] ZB: That's a great question. One fundamental one is just we don't have the legacy that AWS hats right there. Is it an innovator's dilemma? There is this idea that every customer you get who’s super happy with the things that you build suddenly restricts your ability to build something that would be at odds with what they expect.

For example, everything that Amazon has is kind built around this process model where you can lift and shift an existing piece of code and actually run it in whatever environment they provide. When we built workers, we said we’re not going to support, that we support the web platform and we think everyone should just build everything in the web platform. Amazon cannot necessarily do that quite as readily, because they do have so many people who rely on the things that they already built. That’s just the nature of disruption.

We also have 20 million websites that use Cloudflare who really love Cloudflare. One of the things that I love so much about having this job, because before Cloudflare, I had a startup, and at the startup, you build something and then you spend maybe six months working on it or a year working on it and then you spend the next four years trying to get anyone to care.

[00:29:24] JM: That was the first show we did, by the way.

[00:29:26] ZB: Yeah.

[00:29:26] JM: Great show, because I think for the exact reason you just defined. You kind of – I mean, that was a tough startup to build. Tough startup to work on, and I think has a great outcome. Acquired by a company and then a company that kind of had a shared vision and you got to meld with that vision.

[00:29:45] ZB: It was shocking. Yeah. I mean, I don't know anyone's like my company got acquired story that worked out as well as this one did, because it is so amazing getting to work on even more stuff even more quickly and have all these people who really care about the things that we built, because we have all these really great users. That is a big advantage of Cloudflare.

We also have this super dense network, 194 places around the world and not just places in Internet exchanges where you can give them some money and they'll put your servers there, but inside ISPs all around the world. These places that are very difficult to get servers to and very difficult to clear customs in and very difficult to negotiate with and very difficult to place. Those are the huge advantages, because when an engineer walks into Cloudflare and they write a line of code, like 1.1.1.1 is a great example, which is our DNS resolver. It was built by three people in a few weeks basically, and it's the fastest DNS resolver on earth.

Obviously, since then, they've put a lot more time into it, but it would take you and I who knows how long to build the fastest DNS resolver on earth? We'd spend the next 10 years trying to put more servers in more places, and they were able to do it in a few weeks. That's an amazing advantage for everything that we built.

[00:30:56] JM: There also seems to be a willful perspective that Cloudflare has on what it wants out of technology, an ethos. How does the ethos of Cloudflare set the product direction?

[00:31:14] ZB: That's really an interesting question. I mean, I never worked anywhere that had as fundamental of an ethical philosophical core as Cloudflare. I've worked at interesting places that are building interesting technology that I think made good decisions. But to really genuinely want the Internet to be better and make decisions on that basis when people are looking or not looking is a totally new thing for me.

I do believe that it's a huge advantage. In the short term, I think making decisions that are easy can grow a company. One of the things that I've really realized is a lot of the people who work around me at Cloudflare really work there because they believe in what the company is doing. If the company hadn't made those tough decisions when it did and when it does, they just wouldn't be there anymore. It is a core part of the culture that changes everything in a company.

[00:32:07] JM: The other products that you work on aside from the worker's platform, one is access, which is zero-trust networking essentially. We've done some shows on zero-trust networking. So people can listen back to that. But tell me about – I mean, I understand what zero-trust networking is as philosophy for how a company should manage its access to resources. How is it a product? What is the product that you're building around zero-trust networking?

[00:32:42] ZB: That's a good question. If you are traditional company right now, you have a network that has all of your secret internal stuff on it. All the internal applications that someone built that allows you to manage your orders or manage your systems and you go to some IP address and it loads. It's all inside your internal network.

The way that you protect it is by not letting people open up a laptop and connect to your internal network. If people want to travel or leave the office or maybe remote offices, you have a VPN, where it extends that internal network to reach all the way around the world to wherever their computer happens to be.

There are big fundamental problems with that as you know. One is just any device inside the network suddenly connects us everything, and it so easy to get a single bit of malware inside an office building. One exploited laptop or phone is a pretty trivial thing to do.

The question for your organization is how do we move to a model where I am making a decision on a per request basis, whether this person should be able to access thing. The answers a product that we call Cloudflare Access, where you take the Cloudflare Edge that's around the world and you put it in front of all of your services and no one can talk to them directly anymore. They have to talk to this Cloudflare Edge.

The Cloudflare Edge gets to make access control decisions around should this person on this device at this moment with the credentials that they had, which usually looks like logging into Octa, or Google apps for business, traditional log-in just like they would for their email. Should they be able to access this thing right now? If the answer is yes, then the Cloudflare edge allows it to go over a secure tunnel to the actual origin inside your system where the code is. No one is on your internal network, and that's where that word zero-trust comes from. You don't trust anyone. You pretend everyone is the public Internet and everyone has to go through this Cloudflare Edge in order to access your system.

[00:34:34] JM: What are the competing products – You don't have to name specific products, but what are the other ways of facilitating zero-trust networking?

[00:34:46] ZB: Well, that's a great question, because there aren't a ton. It was invented by Google as a part of their BeyondCorp initiative, where they got hacked pretty aggressively and they had to figure out a way to solve this very existential networking problem. They have made something of a business side of it as a part of Google Cloud. If you use Google Cloud, you can use a lot of their BeyondCorp.

There are companies that provide a similar service. The big advantage that Cloudflare had coming into this is, again, that edge network, because when you add Cloudflare in front of your services, we generally aren't going to slow anything down, because your users are already so close to one of Cloudflare's pops. So that's one big advantage.

You have this network density that means you go from a model where all of your traffic is getting VPN to a central location that might be thousands of miles away from where the user is to one where they're connecting to a Cloudflare pop that's within a few milliseconds of them in most places and then immediately able to go to the service that they actually want to connect to. It dramatically improve the performance again, because we're able to take advantage of that giant network that so few people have unless they use workers.

The other big advantage we had is a bunch of technology that we had already built. We have is Argo Tunnel product, which lets us securely make a tunnel from wherever your servers are running your code on earth to Cloudflare so that all that traffic can flow back to your origin

securely. We already had that product and it already worked, and that made it much, much, much easier for us to make this access thing make sense.

We also have Argo, which is a way of routing traffic over the Internet more quickly by making better decisions. Kind of like Google Maps traffic for packets over the Internet, and we're able to turn that on for this as well. Suddenly everyone's ability to clone internal git repo gets much faster. So we're able to take advantage of all these other pieces and parts that Cloudflare has built and make a really powerful system.

[00:36:38] JM: When you decide to build a new product within a company like Cloudflare, where you already have established products, how do you go about spinning up a new team? Do you deprive other teams of resources? Do you put out a job request? Then you have to divide your time among your pre-existing obligations. Tell me about how you spin up a new team.

[00:37:05] ZB: That is the fundamental problem of having a resource-constrained organization, but you want a resource-constrained organization, because you want to be focused and you want to have to make choices. Having an infinite supply of people to work on things slows you down ultimately. The whole Amazon two pizza teams idea. You have to make choices.

Occasionally, there's a product that gets spun down or pivoted or moved in a different direction, and a lot of the time, people help from other parts of the organization. For example, we recently launched a workers HTML parser, and that HTML parser wasn't actually primarily built by the workers team. It was built by the speed team who were building a whole bunch of things to make websites faster, and they needed a new HTML parser and they wanted to be able to use it in workers because it's so much easier to deploy code with workers than the way that we've done it in the past.

So they sat down and built this new parser in Rust and integrated into workers. That required a little bit of support on the workers team, but it was primarily made possible by the fact that people actually wanted to use this thing.

It's similar to how an open source project works, where you have people who want to contribute because they want this thing to be better, because it makes their lives easier. We're able to kind of pull water out of a stone or something like that and get things done that we wouldn't normally have the people or time to do.

The other big thing is only doing stuff that we are uniquely able to do because we have the network and the customers that we have and because we have workers. Access, for example, is totally built on workers, and that means that that team can be a handful of people and write code and it's going to run in 194 places. It makes their life so much easier. They're able to use Argo Tunnel and they're able to use all these other technologies. If they had to build something totally from scratch that didn't take advantage of any of the things the Cloudflare had, yeah, they would need a much, much larger team. So we try not to spend our time on those things.

[00:38:59] JM: Similarly, to what AWS always talks about, they're always building using their previous APIs. They're always building on S3 and EC2 and so on. You have the ability to do the exact same thing, the same computer abstractions that you give to users. You build into a higher-level abstractions.

Another product you work on his WARP. I remember the day that this launched. It was like sitting at the top of Hacker News for the entire day. It was awesome. It was a really good launch page, because I've used VPNs, like mobile VPNs before. Not a great experience. Explain what WARP is.

[00:39:42] ZB: VPN is a useful way to explain it, but I usually say if you know what a VPN is, you shouldn't be using WARP, because most of the people who use a VPN right now are super security conscious or they're super privacy conscious. They are very technological already. They know how to install a mobile VPN, which is pretty difficult for most of these, and they're either trying to access Netflix in a different country than the one they're currently in or they just want more privacy on the Internet.

If you are able to do that, that is awesome. You are wonderful. You are among the .01% of most technologically-adept and privacy conscious people on the Internet. WARP is really for the 99.9%. It's for people who have never heard of a VPN before. They're not looking for anonymity.

They're not trying to do things that are hidden from the site that they're visiting. They're just trying to not be spied on in a coffee shop or to not have their Internet service provider sell their Internet browsing data.

So they need a way to securely encrypt their connection from their phone, which only supports phones for now, but that will grow over time, to the actual website that they intend on visiting. We're trying to give them that. In the process, we're trying to make their Internet better instead of making it worse. Because I would venture that most VPNs make your Internet somewhat worse. That's kind of an established idea that instead of going directly to the place that you're trying to visit, you now have to go to some third-party server somewhere, and that extra hop is going to slow down your Internet by some amount.

We're saying if we upgrade the protocol between your phone and the edge, between your phone and the Internet basically and we use a more modern protocol than what has been used for the last 40 years and we use our Argo smart running technology that we talked about earlier where we know which transit providers and which paths through the Internet are least congested in any given moment and route you that way.

It should be possible for us to make your Internet better while we also make it more secure and more private, and we really believe that that's the only way that we're going to actually change the Internet. That's we're really going to make it a private secure place where no one can spy on anyone, and where it's possible to innovate and move really quickly and make it better.

[00:41:53] JM: This is a consumer product.

[00:41:54] ZB: It is the first Cloudflare consumer product. Yes. It is meant for your parents and your friends, people who use the Internet but they're not super technical. They certainly don't have a website, and you install an app on to your phone. It's called 1.1.1.1, and we send all of your Internet traffic over WireGuard, which is a super modern, very secure, basically, VPN protocol, to Cloudflare's Edge before it goes out on to the Internet. You have this secure, fast, optimized, more modern protocol tunnel between your phone and the world.

[00:42:27] JM: Why does that differ from the naïve way of sending Internet traffic?

[00:42:33] ZB: That's a great question. There are two protocols that are really commonly used to send data over the Internet. One is called TCP and one is called UDP. TCP is a really robust, very complex protocol that guarantees it will send the thing that you intend on it sending in the correct order with the correct content to your destination.

The problem with TCP is it has a lot of overhead and it doesn't handle the key things that commonly happen on the mobile Internet very well. When packets get dropped, it forces everything to basically grind to a halt, retrieves those packets, gets them back in order and ensures that you have this really, really reliable connection between two points.

It turns out though the last 40 years we figured out a lot of things that allow us to do this in a more robust way, more efficient way. So instead of needing to grind everything to a halt and reconstruct packets and sort out the order, we can send things much more quickly and figure out the things that we missed later on and reconstruct them and use more modern protocols in order to make this happen.

A lot of this concept is very similar to what's being done with HTTP 3 right now, where the world is deciding we don't want TCP to manage the reliability of our connections. Just give us UDP, which is much less reliable. It doesn't provide any of those guarantees protocol, and we'll do all of that other stuff on top of it. That's a part of what powers WARP.

[00:44:00] JM: Any benchmarks and how much faster HTTP 3 three is than HTTP 2?

[00:44:05] ZB: There are. I don't have them. You can actually try it yourself, because Cloudflare shipped to HTTP 3 recently. So you can turn it on for your website. With Chrome Nightly, I guess, or a very modern version of Chrome, you're actually able to browse the Internet with HTTP 3 now.

[00:44:20] JM: The security is improved, because – I guess I'm not quite getting the security component. So why is the security improved?

[00:44:32] ZB: Your data is now public key encrypted from your phone through your local cell tower or Wi-Fi network or wherever you are all the way through your Internet service provider so they can't see anything that you're doing all the way to Cloudflare's Edge. That's very important for two reasons. One is if you visit a non-HTTPS website, a non-secured website, everything you're doing is in the plaintext. They're able to get the content of the site. Anything you submit to the site. Very, very insecure, very bad. But even if you are visiting HTTPS website, because of something called SNI, it's actually possible to see all of the websites that you visit because you have to tell the server what website you intend on visiting before it can negotiate the TLS secure connection.

Your ISP right now or anyone snooping on you in a coffee shop Wi-Fi knows every single website that you go to, and maybe that's a big deal to you. Maybe it's not, but it certainly is a limitation on the privacy guarantees of the Internet. No one that I know feels super confident in the fact that their ISP is not selling all of the data.

I would say, if there's money to be made in it, they probably are. This idea that we can encrypt this big chunk of the transmission, the least secure chunk of the Internet transmission, greatly improves the privacy and the security of people using the Internet.

[00:45:56] JM: It doesn't cause any kind of problems with the endpoint I'm trying to hit, right? If I am authenticated with Google and I'm trying to access my Gmail, it's not going to cause any kind of issue.

[00:46:10] ZB: Tight. It's going to go through CloudFlare's Edge, but it's still going to go to Gmail and we're going to do our best to inform Gmail of who you are. Again, we're not trying to provide anonymity for anyone, but it will go over that encrypted tunnel.

[00:46:23] JM: What have you learned building a consumer product?

[00:46:29] ZB: That is a great question. It is so different. One thing is just the volume of support requests. I mean, in the app, we have a button you can push to send us a ticket. You could send us feedback about the app or something like that. When you're deploying something that runs

all around the world and is victim to all of the different Internet connections, in all of the ISPs everywhere, there are all sorts of issues for us to work out. We like that.

That's one of the reasons we launched WARP consumer products, because we want to work out every issue and become really, really good at securely tunneling Internet traffic. But there are a lot of support requests and there a lot of people who have questions or concerns or want to talk to us or want to tell us that it didn't work in a certain situation in a certain moment. The volume of that is definitely a new thing for me to experience.

Another is how happy and nice people are. When you're offering someone an app that's mostly for free, you can pay us a little bit of money and we'll use that art or routing technology to make your Internet work a little better. But in general, it's free. People are really forgiving and they're really kind and they're really nice and it's amazing how many people you can run into that really believe in the mission of 1.1.1.1 and WARP in a way which I totally didn't anticipate.

[SPONSOR MESSAGE]

[00:47:54] JM: If you want to extract value from your data, it can be difficult especially for nontechnical, non-analyst users. As software builders, you have this unique opportunity to unlock the value of your data to users through your product or your service.

Jaspersoft offers embeddable reports, dashboards and data visualizations that developers love. Give your users intuitive access to data in the ideal place for them to take action within your application. To check out a sample application with embedded analytics, go to softwareengineeringdaily.com/jaspersoft. You can find out how easy it is to embed reporting and analytics into your application. Jaspersoft is great for admin dashboards or for helping your customers make data-driven decisions within your product, because it's not just your company that wants analytics. It's also your customers.

In an upcoming episode of Software Engineering Daily, we will talk to TIBCO about visualizing data inside apps based on modern frontend libraries like React, Angular, and VueJS. In the meantime, check out Jaspersoft for yourself at softwareengineering.com/jaspersoft.

Thanks to TIBCO for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:49:31] JM: As we near the end, I want to revisit serverless a little bit. I don't know how much you've looked into the other models of serverless. Have you looked much at Fargate, or Knative, or Firecracker? Do you have any perspective on these different technologies?

[00:49:52] ZB: I can speak generally. A lot of that is about saying we have a VMWare, we have a container we want to make incrementally faster.

[00:49:59] JM: Yes.

[00:49:59] ZB: That's a super valuable business to be in, because if you can be 100 milliseconds faster than someone else, then customers will choose to use you. Since this is already an established market, a lot of people buy VM's every day. There's a ton of money that you can make really quickly just by being the slightly faster version of the things someone's already looking for. That is very different than what Cloudflare is doing.

Cloudflare has been a more disruptive place where it's saying, "We're going to build something that doesn't work the same way as the things you already use that you're going to have to architect your applications around a little bit." In return, you might get something that's 100 times faster than what you were doing before instead of two times faster. I see them as very different, totally different world appealing to totally different customers, totally different maturity level of the market. Awesome products, awesome technology, very, very impressive engineering.

[00:50:50] JM: Has Cloudflare gotten into open sourcing much of its technology or does that just not make sense because you are in a pretty distinct position?

[00:51:01] ZB: Cloudflare does open source a lot of stuff. For example, we were talking about WARP, and our Rust WireGuard implementation is totally open sourced. We would love for people to use it and think everyone should use it.

We tried to open source as much we can. Obviously, it is a process and it takes a lot of work and there's a lot of overhead to managing and maintaining an open source project. We're pretty confident doing it around libraries and things that are low-level that only a limited number of people can use, but doing it for like complete product level things is a little bit more challenging from a resource constraint problem space, but we like open sourcing things. It's fun and we use a lot of open source things.

[00:51:40] JM: Given how much you think about product strategy, I think you have a pretty good perspective on how to think about competition. I've had a lot of conversations recently with companies that are competing with cloud providers from a different angle than you. You're competing with cloud providers directly. I mean, you're CDN. Most of the cloud providers offer their own kind of CDN. I don't know. Maybe you wouldn't use the term CDN edge computing.

But Cloudflare has managed to differentiate and build a gigantic business. There are these other companies that have kind of a co-op petition with AWS where they have Elasticsearch or a database product and they feel like there is an uncomfortable adversarial relationship. Do you have any reflections on how to think about competition in the modern era of software infrastructure?

[00:52:42] ZB: Well, I think I would be very afraid of those types of relationships with Amazon as well, because you look at like Toys R Us where they outsource their fulfillment to Amazon and then basically went on a business. It is certainly a dangerous thing to get into a close business relationship with the enemy in that way, if that's how you relate to them.

I think, for us, there are just so many more people on this earth who have never heard of serverless than who have heard of serverless and are using a competitor. The mission for all of us really is expanding the market greatly to usher in a new era of how you build software, and we're so far away from completely succeeding with that. We're really only reaching the earliest of the earliest adapters right now and we're giving them kind of a halfway tolerable experience, but certainly not the level of the experience that should be possible with the technology itself, but will be possible overtime.

So much of this mission is really about making developers happier, making them have a better experience using serverless. Make more people know what serverless is and want to use it and want to answer questions like which provider they should use, and that represents I think 99.9% of the growth available in this world, and competition between vendors right now is .1% of the available money.

[00:54:03] JM: Last question. How will cloud providers look in a decade?

[00:54:08] ZB: That's a great question. I hope the answer is not one. I hope the answer is not there is one giant successful cloud provider through network effects has been able to capture the entire world. I think the world is more fun with disruption and with companies succeeding and then failing and ebbing and flowing, and it seems like innovation happens faster and things get better for developers more quickly. That's really what I care about.

When there's a lot of competition and a lot of things are churning, I know that gets on people's nerves, particularly in like the frontend ecosystem where they feel like every year they have to learn a totally new stack. But I look at how much better it is to develop code on the frontend now than it was 10 years ago and I'm like, "Thank God!" Even if you just stop now, your life is so much better than it was. I hope that continues to happen. I hope I look back in 10 years and the way that you build a website is you go like I want Facebook, but for cats, and you hit enter and that is – Developers get to think about entirely new classes of high-level problems than they can worry about today.

[00:55:11] JM: Actually, I allied. One more question. When you were working on your company, I think it was 2014 through 2016, right? What was the name of the company?

[00:55:19] ZB: It was called Eager.

[00:55:20] JM: Eager. How has the world diverged from where you thought it was going to go in that time? Because in Eager, you are making a specific bet on a view of the future. How has that view of the future been diverged? What has come out differently than you anticipated?

[00:55:41] ZB: I would say that everything happens somewhat disappointingly slowly. You really hope that sitting in 2014, the world of being a developer or the world of having a website would be dramatically different in 2019, and it isn't. It really isn't.

[00:55:57] JM: Plugin ecosystem is pretty much the same.

[00:55:59] ZB: Right. Maybe you use React instead of using Angular, but the experience of writing code and how you get the code that you use and how someone installs some code and how they add things to a website or make them better, all of these pieces really work exactly the same.

The answer that I'd give is like I am surprised at how long it actually takes for things to really change. I see a lot of what we're trying to do at Cloudflare to be making it such that everyone can really use the Internet. Everyone can build things on the Internet. Not just a tiny set of super technical people. When the Internet started, it's just being used by a handful of researchers and it has gradually expanded to the point where if you have a coffee shop, there's a way that you can make a website with limitations.

I really see this journey as ending when every single person can use the Internet comfortably to build things and create things and interact with things and not get hacked and not be in danger, whatever they want on any device. I think we're still pretty early. I think the tools still require you to have like years and years of experience before you can really safely build things on the Internet. There are so many ways to shoot yourself in the foot that like you have to be very, very skilled and experienced to not like just security vulnerabilities, for example. I just see it all happening much slower than I expected.

[00:57:22] JM: Zack, thanks for coming on the show.

[00:57:24] ZB: Yeah. Thank you. This was really great.

[END OF INTERVIEW]

[00:57:34] JM: Software Engineering Daily reaches 30,000 engineers every week day, and 250,000 engineers every month. If you'd like to sponsor Software Engineering Daily, send us an email, sponsor@softwareengineeringdaily.com. Reaching developers and technical audiences is not easy, and we've spent the last four years developing a trusted relationship with our audience. We don't accept every advertiser, because we work closely with our advertisers and we make sure that the product is something that can be useful to our listeners. Developers are always looking to save time and money, and developers are happy to purchase products that fulfill this goal.

You can send us an email at sponsor@softwareengineering.com even if you're just curious about sponsorships. You can feel free to send us an email with a variety of sponsorship packages and options.

Thanks for listening.

[END]