EPISODE 978

[INTRODUCTION]

[00:00:00] JM: Amazon EC2 or Elastic Compute Cloud is a virtualized server product that provides the user with scalable compute infrastructure. EC2 was created in 2006 as one of the first three AWS services along with S3 and simple queuing service. Since then, EC2 has provided the core server infrastructure for many of the companies that have been built on the cloud.

A large scale virtualization product like EC2 requires its engineers to have a deep understanding of scheduling and multitenancy. In previous shows, we've touched on subjects such as hypervisors, the noisy neighbor problem, the cold start problem and other aspects of multitenant infrastructure. To make EC2 successful, these issues must be continuously revisited and resolved at different areas of the stack.

Dave Brown joined the EC2 team in 2007 and now leads the EC2 compute, networking and load balancing teams as a vice president. Dave joins the show to discuss the history of EC2 and the canonical problems of virtualized server infrastructure.

We are hiring a software engineer who can work across both mobile and web applications. This role will include work on softwaredaily.com, our iOS app and our Android application. We're looking for someone who learns very quickly and can produce high-quality code at a fast pace. We're looking to move beyond the world of just being a software podcast into more of a platform of information about software.

If you're interested in working with us, send an email to jeff@softwareengineeringdaily.com. We're looking for somebody who is hungry and wants to learn quickly and wants to build lots of software. If you are that person and you're hungry, it doesn't matter what your experience level is, as long as you have built and shipped meaningful applications. Send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:02:04] JM: If you are a SaaS or software vendor looking to modernize your application distribution to gain more enterprise adoption, checkout replicated.com. Replicated provides tools to deliver your Kubernetes-based application to enterprise customers as a modern on-prem private instance. That means your customers will be able to install and update your application just about anywhere.

Bare metal servers in a cloud VPC, GovCloud and their own Kubernetes cluster, vSphere. This is a secure way the your customers can use your application without ever having to send data outside of their control. Instead of your customer sending their data to you, you send your application to your customer.

Now, this might sound difficult and maybe you're not used to it because you're a SaaS vendor. You're a software vendor, but Replicated promises that recent advancements from tools like Kubernetes make it far easier than before, and the Replicated tools can help vendors operationalize and scale this process.

The Replicated tools are already trusted by noteworthy customers like HashiCorp, CircleCI, Sneak and many others. As a result, over 45 of the Fortune 100 already have an application deployed via Replicated in their infrastructure. That's a strong sign of adaption.

Go to replicated.com for a 30-day trial of the full Replicated platform. You can also listen to an interview with Grant Miller, the CEO of Replicated, that we did a while ago.

Thank you to Replicated for being a sponsor of Software Engineering Daily, and you can check it out for yourself at replicated.com and get a free 30-day trial.

[INTERVIEW]

[00:04:12] JM: Dave Brown, welcome to Software Engineering Daily.

[00:04:14] DB: Great, man! Thanks for having me on.

[00:04:16] JM: You were one of the original laborers on EC2. EC2 was originally built on the Zen hypervisor. What kinds of modifications did you have to make to Zen to fit the spec for what you wanted out of EC2?

[00:04:29] DB: Yeah. When we first started building EC2, one of the last thing we actually did was really look at modifications of the hypervisor. There was a lot to do. How do I build a distributed system that could actually run virtual machines at scale, take requests from customers? Remember, back then there was nothing like it really on the internet that allowed you to host machines and run service.

We kind of wanted to simplify things initially, and so we just took pretty much standard Zen and used that. That was what we launched back in August of 2006. As we sort of over the next few years, we started to make few modifications. So there were a number of things we did across Zen. One of them was run the scheduler as sort of scheduling. The largest changes we did there was for our T1 instance type when that came out, which is an instance type that allows you to burst. Most of our instance types today give you – We kept the capacity. You get exactly what you've asked for. Without T-series, you're actually able to get – Be able to burst. So we did a lot of changes to the Zen scheduler there.

The other area we've invested massively in Zen has been around how do we do security updates and be able to perform updates to the hypervisor without actually impacting customer workloads. For the most part when update Zen you would normally have to affect the customer workload or reboot the machine. We've actually been able to improve Zen and make changes that allows us to do those updates what you call live update where it never impacts the customer workload at all.

Some of those things have now actually made their way upstream into the current version of Zen. Obviously, we still have a large Zen fleet. Nitro is our new hypervisor. We still have a lot of the Zen instances. Some of those things have made their way upstream now as well and we work very closely with the Zen community. Initially, not much, but over the years, obviously, we've changed quite a bit.

[00:06:05] JM: How do you choose between making a change in Zen itself versus maybe building some kind of new system or a module or a scheduler on top of Zen?

[00:06:15] DB: I suppose it just really comes down to what needs to be done to actually get the job done. What are you going to do and what's the most efficient thing to go and where to make the change. With like the scheduler, there's really no other place to do that. With the security updates, no place to do that.

Ultimately what happened overtime is we've realized that we needed to move a lot of our components to hardware, because we knew that we would get much better performance there than continuing to run with inside a software hypervisor. That's where we ultimately went and built a lot of components outside of Zen and both on hypervisor, which ultimately became the Nitro system.

[00:06:47] JM: What was the hardest engineering problem in starting to build hardware-based hypervisors.

[00:06:54] DB: We got into it sort of relatively slowly. I mean, it's an enormous project. So when you think about taking the hypervisor for EC2 and replacing that, that's a multiyear project and it's literally changing almost every single component both within the EC2 data plan and across the EC2 control plane.

Where we started was in 2012. The first thing we wanted to address was to get better network performance. Back then we were struggling with sort of tail latencies on network. So the network latency itself wasn't bad, which we get from a software hypervisor like Zen. The tail latencies were a challenge for some customers. What I meant by that is you see jitter. So you'd have sort of a baseline of latency, but then you'd get these spikes every now and then of latency that we couldn't find a way to solve. That just comes down the fact that our hypervisor was sharing the same physical core, the same server as the customer workload. Anything that's putting a load in that core is going to affect other components and can give you jitter.

That's where we decided back then was to offload the networking to a physical card. We essentially got a card we could run Linux on, and we wrote our networking stack to run on that

card, and that became part of the NICs. The network packets would come into the NIC, with the whole entire networking, the processing that we need to do, which includes encapsulation of those packets to make sure that it can work within a VPC, and we'd run our own software-defined network there and there are a lot of stuff that has to happen before it gets to your instance. That all happened on hardware.

We launched that with our C3 instance at Reinvent in 2012. So our first Reinvent actually. That have massive improvement in network latencies. That Jitter we were speaking about just completely disappeared. That was the sort of first sign that this hardware was the right path to go.

Then from 2012 through to end of 2018 or 2017, sorry, which is when we launched our C5 instance. It was really this ongoing process of offload. We started with networking and we did EBS, so offload for EBS storage. Then we did NVME drives, and so control for NVME drives with our I3 instance. Then eventually we said, "You know what? This has moved everything. Can we get to a place where the actual core that the customer is going to use, the processor that we make 0% of their core."

If you go back to what we had on Zen, about 20% of their core was dedicated to EC2 processing, and 80% given to the customer. Where we're on today is 0% is dedicated to EC2 and 100% of their core is given to the customer. That's why we're able to do things like bare metal. Bare metal is essentially running an EC2 instance without a hypervisor. We're able to run our entire Nitro system and then remove the hypervisor completely. So you literally have bare metal access to the underlying hardware, which has been quite a big step forward.

It was a journey. I think one of the biggest challenges was just honestly committing to that journey and the amount of time you think about it as a business and you literally want to rewrite honestly every single component. It's quite daunting. I'm very happy we did it. I think it's given us an incredible advantage, and our customers. It's just an amazing performance obviously to instances today.

[SPONSOR MESSAGE]

[00:09:55] JM: Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:11:45] JM: The engineering challenge is that you've seen evolve with the popularization of containers. How closely have those engineering challenges mirrored what you saw prior to that with the maturity of virtualization?

[00:12:06] DB: I'm not sure I would draw many parallels there. I think virtualization back in the day, I think everybody just accepted that it wasn't going to be as performant as running on actual hardware. There was a big challenge there.

I don't really think that's translated a lot into containers. I think where we struggled more of how to think a little differently about containers and obviously the serverless space as well is the rate of change, right? If you think about where we started when we first started to virtualize, you would maybe launch on instance on VMware or some other virtualization system and you'd run that instance with some period of time.

When EC2 came around, you would launch an instance and run for a period of time. You would shut it down. Your instances became a lot more immutable. I think in the container space, when you get to the serverless space, things are just becoming more and more immutable. You launch them, you run them for a few seconds because you need to have a task and you shut them down as well.

That's where we've had to think a lot in terms of the hypervisor and how much time does it take us to actually boot an instance, for example. If that takes too long, you can't run these sort of very ephemeral workloads that you'd want to be running with containers and serverless transactions. Hypervisors changed a lot to become a lot faster to be able to support container workloads.

But then also things like Firecracker, which is a new hypervisor we built specifically for serverless and container workloads is sort of built on top of Nitro hypervisor but allows you to launch some machine in literally a few hundred milliseconds. That's allowed us to even support more of these ephemeral workloads. I think you're going to continue to see that workloads are going to be more ephemeral where you don't have a machine learning until you absolutely need to sort it up, make use of it and shut it down again, and that's really sort of where the container space is going as well.

[00:13:46] JM: The cold start, are you basically saying that the cold start problem was not as much of a big deal when people were more focused on virtualized server infrastructure because those VMs were longer lived than the containers?

[00:14:05] DB: Absolutely. I think back in the day, VMs did run for a longer period of time. People tended to launch a machine and make use of it. They weren't thinking about this idea of launching machines and shutting them down when you didn't need them anymore.

I actually remember the first time I saw a customer really do that at scale I believe was back in 2008. I remember it was a Thursday and there was this company called Animoto. Do you remember them? I think it's called Amimoto, Animoto, and they had this thing where you could basically – I think it was you point them at your Facebook stream and they would generate like a video of – If you're to music with crazy graphics and interface. It's this cool little gimmick.

On a Thursday afternoon, because I remember, they started doing this thing where they would launch a new instance for every movie that somebody subscribed too and they went viral. We certainly saw this massive spike of launches on EC2, and we kept it up, but I won't lie. I'm not lying when I say there were a number of us behind-the-scenes making sure that then early EC2 wasn't going to fall over with a sudden increase in load.

[00:15:01] JM: They're spinning up a new VM for each video?

[00:15:04] DB: Yeah, we'll spin up a new VM for each video, which I think we later found out it was actually a bug in their code, but that was the first time that we actually saw this idea going from literally almost very little capacity to thousands and thousands of machines suddenly starting up. That was the first time we saw it. Obviously we've seen VMs load for shorter periods of time. Then when you get to serverless and the cold start thing, you want those Lambda functions to start up instantaneously.

We've been working a lot as EC2 with the Lambda team, the secret, we shouldn't tell anyone, but serverless actually has a server behind-the-scenes. That server for Lambda is EC2. We've been working with them to solve the cold start problem and there's really two things we had to do. One was give them a hypervisor that has a very, very fast start time. That was Firecracker, which we actually ended up open sourcing as well.

You can go look at Firecracker out there, and it's really designed to be incredibly lightweight. Give you the same security boundaries that we have with the Nitro VM, which is something we've never compromised on. We'll never compromise the boundary between customers VMs, right? We never want to take any changes there, but can start up a VM in a couple of hundred milliseconds, 100, 200 milliseconds a time. That solves one part of the problem of Lambda.

The other part of the problem was for customers using VPC with Lambda, how quickly can I instantiate the networking resources? How long does it take to attach an ENI, for example? We've done some work with the Lambda team to get that down to a couple of milliseconds as well. I believe Lambda is actually – They've announced and they've rolled out to most of their fleet sort of a solution to cold start. Lambda functions are now starting a lot faster and supporting those very ephemeral workloads again, which is where the world is going.

[00:16:37] JM: How does the Firecracker-based vision for serverless infrastructure compare to – I don't know if you're familiar with this stuff, but like the Knative suite of projects from the Kubernetes community?

[00:16:52] DB: Yeah. I don't know too much about Knative, but we have been working a lot with a number of those vendors. What happened with Firecracker is we built Firecracker and we did use a number of those tools. Not Knative. The name of the other one, I'm losing it right now. I'll remember shortly. But we ended up building it and launching it and it was also at Reinvent last year. It was the number one GitHut project for two days. It's pretty insane. We couldn't believe it. It just showed the desire and the community with something in that space, right? We've been very excited to contribute there from an open source point of view.

My Firecracker team, which is actually based in Bucharest, Romania, I have an amazing team up there that built Firecracker. They've been very engaged in the community. So a lot of those teams are working together. I'm not 100% sure if Knative is involved. But then we've also had another project called rust-vmm, which is sort of building a VM management system on Rust, which obviously everybody loves Rust at the moment. There's a lot of engagement there.

Again, with these serverless project, what's been really great is sort of the whole competitiveness kind of goes out the window and it's all community coming together and saying,

"Hey, what can we actually build?" Whether it's ourselves as the cloud provider, any of the other cloud providers or any of the other chip manufacturers and that sorts of things all coming together and working on that project. Early days on all of that stuff and we're excited to see where it goes.

[00:18:10] JM: Does Firecracker mean hard dependencies on specific hardware?

[00:18:15] DB: I don't believe so. Obviously, I think at the moment – I believe we've just finished one [inaudible 00:18:18]. It's an x86 processor and it's obviously a hypervisor set as run on bare metal. You can run it on – We have a lot of customers that run it on EC2 bare metal today. But there are also customers that run it themselves on any other hardware that they'd like to use. We've had people even look at using it in like small devices, embedded devices and things like that. It is very, very lightweight. If you need sort of virtualized environment, it's something you can definitely look at.

[00:18:44] JM: The workflow for using Firecracker is you spin it up on an EC2 instance and then you spin up your own serverless infrastructure on top of it or are you spinning up AWS Lambdas on top of it? Can you help me understand what the use case is?

[00:19:01] DB: The way that it works with Lambda is Lambda uses Firecracker, but you don't actually see Firecracker when Lambda uses that. Your interface is talking to Lambda and starting up a – Creating a function and then executing the function.

What Lambda is doing behind-the-scenes is they're taking a bare metal EC2 instance and they're installing Firecracker on to that instance and then they manage it in Firecracker as if it was a VM. Much the same way you would have done with Zen, they're creating their necessary VM instances or containers, whatever you want to call them, [inaudible 00:19:30], within Firecracker.

So, they could put thousands of machines, servers instances on that physical machine, and each one of those will be tied to a customer function. When the function executes, it's running inside that Firecracker VM. That's how you would manage it. If you run Firecracker yourself, you would download it from GitHub and then install on the machine as a hypervisor essentially, and

that's the same way as Zen does, and then interact with the Firecracker as a set of APIs that allow you to create images and instances and use it as a hypervisor.

[00:19:59] JM: Okay. Firecracker is a hypervisor.

[00:20:02] DB: Yeah.

[00:20:03] JM: Okay. Got it. What went into the design that allowed you to spin up images faster and reduce that cold start problem? What were you able to strip away or what kind of performance areas were you able to improve?

[00:20:17] DB: Yeah. In the case of Firecracker, it's making incredibly lightweight, and in reducing the number of devices that you actually emulate. One of the challenges is –

[00:20:25] JM: No USB.

[00:20:26] DB: Probably no USB. There's certainly no printer. What happens to a lot of these hypervisors is they emulate pretty much early devices being out there in the past, right? If you look at what's happening with Zen and QMU, it's just a massive lot of time in the emulation. A number of other optimizations as well, I've done all the details on all of them, but it's really stripping that away and getting it to boot. Obviously, making sure that the image you're booting can also be started out very, very quickly. Thinking about the operating system and what are you bringing into memory there?

The other side of it was just making sure that the network performs a lot faster and at the state of VPCs being pushed out a lot faster. That got us down into the sub-second time range for cold start times on Amber.

[00:21:04] JM: That seems like a really good approach to the cold start. There's also an approach I've heard some people talk about where you preload or pre-warm a bunch of containers with like Node.js or with Python on it so that as soon as a workload comes in that requires Python on a container, you can just schedule that workload on to the container that's

pre-filled with Python, and boom! You run it really quickly. Do you think that's also viable approach to reducing the cold start?

[00:21:38] DB: It's certainly a process we've looked at, and we've definitely done pre-warming in other parts of EC2 and AWS, right? There are a number of services that will pre-warm servers and use them. It depends a lot on – There are a couple of things you want to think about there. One is the cost. How low does your pre-warm pull need to be? Because you're essentially keeping capacity around that you may use at some point in the future.

You want to have enough capacity, because when you don't have capacity in your pre-warm pool, you end up with a slow list start time if you're using that model. But if you have too much capacity in your pre-warm tool, you're spending money that you shouldn't be spending. That's one of the things.

The other one is from a security point of view. Where is the security boundary? When you have a pre-warm pull, there's going to be nothing in that pre-warm pull that you wouldn't want to give to any random customer. When you spin that machine up, is it ready for that customer or is it a pre-warm pull and then allocated to a customer?

You're also going to think about with your account boundaries. You can't move a machine between accounts. So if you do have a service where each uses actually on a different AWS account, it doesn't work in that arena. There are a number of things – Many services have used pre-warming. One of the teams that's in my organization is the Elastic Load Balancing team as well, and they had used pre-warming. We've used pre-warming on Elastic Load Balancing for some of our older load balancers where it also runs on EC2, and when we need to have a new machine or a new node for the load balancer, we're able to pull it from our pre-warm pull, and then you really avoid the boot time. You're able to get another machine into a service a lot faster. There's definitely a place where we've used that very effectively. I'm sure our customers do it as well.

[SPONSOR MESSAGE]

[00:23:12] JM: As businesses become more integrated with their software than ever before, it has become possible to understand the business more clearly through monitoring, logging and advanced data visibility. Sumo Logic is a continuous intelligence platform that builds tools for operations, security and cloud native infrastructure. The company has studied thousands of businesses to get an understanding of modern continuous intelligence and then compile that information into the continuous intelligence report, which is available at softwareengineeringdaily.com/sumologic.

The Sumo Logic continuous intelligence report contains statistics about the modern world of infrastructure. Here are some statistics I found particularly useful; 64% of the businesses in the survey were entirely on Amazon Web Services, which was vastly more than any other cloud provider, or multi-cloud, or on-prem deployment. That's a lot of infrastructure on AWS.

Another factoid I found was that a typical enterprise uses 15 AWS services, and one in three enterprises uses AWS Lambda. It appears serverless is catching on. There are lots of other fascinating statistics in the continuous intelligence report, including information on database adaption, Kubernetes and web server popularity.

Go to softwareengineeringdaily.com/sumologic and download the continuous intelligence report today. Thank you to Sumo Logic for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:25:02] JM: Inside of Amazon, you, to some extent, get a preview of what the rest of the world is going to be seeing in a decade. You see this in internal marketplace things that people are building as well as, of course, in AWS, services that are in beta or services that are in alpha, or whatever, pre-alpha. I know you can't give too much secret sauce. Can you give me some like broad predictions or areas, things that I should keep my eye on that in 10 years are going to seem as inevitable as serverless or an edge computing?

[00:25:38] DB: Yeah, 10 years is a long time.

[00:25:40] JM: Okay, 5 years. Whatever your prediction horizon is.

[00:25:41] DB: Three months. Is three months okay? I'm not sure.

[00:25:43] JM: Sure.

[00:25:45] DB: We could look at it. I mean, one of the things to think about is just how much have things changed in the last 10 years. EC2 started 14 years ago. When I joined the EC2 team, we had no idea what this thing would be. Where it is today and where the cloud is going today, I think things have changed a lot so. Things change an enormous amount in 10 years.

I think some of the interesting things, we already spoke a little bit about the very ephemeral workloads, right? Things, you're going to be spinning up machines and using them for shorter periods of time. Serverless obviously is going to be a massive movement in that direction. I think some of the other things we're excited about is we launched our own processor at Reinvent this year. Last year we put an Arm processor out there that was what we called our AWS Graviton Processor. We had the A1 instance and was really just to test in the market and putting something out there and saying, "Hey, here's an Arm processor. You just see what you do with it."

Arm has been very big in the mobile space but hasn't really had a big play in the server space. We were very, very happy with what happened with Arm, because it provides you – The initial one provided you with 45% cheaper price performance for your workloads. A workload that ran on an [inaudible 00:26:47] processor, if you could put it to Arm, you could theoretically say 45%, which is a massive saving.

What we announced this week at Reinvent is the Graviton 2 Processor, and we've actually been working on this for a number of years now already, and we're very excited about the performance we're seeing. It's our very first 6 generation instance type and giving us significantly better performance than what you can get on current x86 processors. It's just a leap forward in processor technology.

If we think about where we're going to be going over the next couple of years, it's really up to the community now, up to our customers and engineers and developers and software engineers

to think about how are they going to be – Are they going to be putting to Arm? Is Arm really going to become a very viable server chip? I think we've put one out there now that is really a leading chip in many ways, so it'll be interesting to see what engineers do with that.

I think obviously more and more stuff is going to continue and move to the cloud. I think one of the big challenges we're having now that I think I'm thinking a lot over the next couple of years and be very interesting to see is latency and networking. What's really happened is if you think about how far we've come over the last 10 years, everything basically is improved to levels that removes a lot of blocker for our workloads. CPUs have now got into a point where they're very seldom the blocker for any sort of workload anymore. [inaudible 00:28:05] died about 4, 5 years ago. So we're really at an amazing place from a CPU point of view.

Memories made an incredible progress. NVME drives and SSDs have just made an outstanding performance. The one thing that I think is limiting some workloads and what I'm starting to see is this need for lower latency. The world is now more mobile. A lot of the stuff we're consuming from our phones or the LTE networks. You think about autonomous vehicles, you think about where IoT is going, you think about robotics and what they're doing in data centers and machine learning vision, you've got things happening at a very high speed where you need to make decisions and do inference and latency is becoming a problem.

What it comes down to is the speed of light. We're still working on solving that problem of can we make it go any faster? It's a joke. I don't think anybody can. That's where you've seen us do things like, "Well, what do we have to do to solve this problem of latency?" Well, we've got to move our workload closer to our users and closer the way our customers use us all.

We spoke about a number of projects this week at Reinvent as well. One of them being Outpost, which allows us to bring our hardware into your data center. We also spoke about local zones, which is this idea of being able to put an AWS availability zone close to our user base. I'm excited to be launching our first one there in Los Angeles, which is actually going to be supporting the movie industry.

What they're doing there is filming movies or TV shows on a daily basis. The content, which is [inaudible 00:29:24] shot much higher resolution than 4K. That's uploaded into AWS and then

actually have designers and editors and animation artists working on it on EC2, on our G4 instance, through the night to make sure that it's ready for the next day and then the next. This crazy high velocity production process. So we're putting – Latency there is very important, because you can't do animation without sub-ten millisecond latency. Our region in Los Angeles has given that industry access to very low latency compute.

I think one of the things that's going to have the biggest impact is 5G, and 5G I think is not just the next version of 4G. It has a number of different networking features, but two of them is obviously much, much, much higher throughput. When you get a couple of tens of megabits from your current 4G connection, you could theoretically get up to gigabytes, 1 to 5 gigabytes from a 5G connection from your mobile device, which is just crazy. Then the latencies is going to go down to single digit milliseconds. That completely changes what we can do from an engineering point of view.

If I'm suddenly able to have a mobile device that can get that sort of latency, suddenly I can probably really do games that are hosted server-side and have a really sort of client-side experience with that, right? Autonomous vehicles can now upload data. Potentially even make decisions whether I should brake or not, but decisions without a remote to that vehicle maybe pause for Audi and things like that.

Driving factories for example, could people get rid of Wi-Fi completely and really just move to 5G? So it just becomes this sort of ubiquitous connectivity that's going to be everywhere. I don't think we know what low latency is going to give us in terms of software applications. I don't think we knew what 4G was going to give us and what the iPhones and Android phones are going to give us and how much they're going to change over the last 10 years.

I think that's sort of the thing that we're really watching, is getting latency down, and it's going to drive a whole lot of new workloads, and I think we're all going to be very, very hopefully pleasantly surprised about how that changes our lives and what new software is going to be out there.

[00:31:18] JM: Dave Brown, thanks for coming on the show.

[00:31:19] DB: Fantastic. Thank you very much.

[END OF INTERVIEW]

[00:31:30] 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.

[END]