EPISODE 1224

[INTRODUCTION]

[00:00:00] JM: Software-defined networking describes a category of technologies that separates the networking control plane from the forwarding plane. This enables more automated provisioning and policy-based management of network resources. Implementing software-defined networking is often the task of a site reliability engineer or SRE. Site reliability engineers work at the intersection of development and operations by bringing software development practices to system administration. Equinix manages colocation data centers and provides networking, security and cloud-related services to their clients. Equinix is leveraging its status as a market leader in on-prem networking capabilities to expand into cloud and infrastructure offerings such as Equinix Metal, which has been referred to as bare metal as a service, and offers integrations with third-party cloud technologies with a goal of creating a seamless alternative to modern public clouds for organizations seeking the benefits of co-location.

Tim Banks is an engineer with Equinix and he joins the show to talk about what Equinix offers and how it differs from other cloud providers.

[INTERVIEW]

[00:01:04] JM: Tim, welcome to the show.

[00:01:05] TB: Thank you. Thank you, Jeffrey.

[00:01:07] JM: There are a variety of modern cloud providers these days. You've got gigantic cloud providers like AWS and GCP. You've got the edge cloud providers like Cloudflare. And you've got companies like Equinix, which is a different sort of cloud provider. Can you explain how Equinix compares to the edge cloud providers and the big mainstay cloud providers?

[00:01:34] TB: Sure. I think the biggest difference that you see with Equinix is that the fact you're still you have in essence your own hardware at Equinix. There's no multi-tenant. There's no hypervisor running underneath there. You're not borrowing a container. You're not borrowing network or anything like that. You literally have metal systems, servers of your very own that you're not sharing with anyone that whether if you're using metal, get provisioned by API, you can colo or whatever, and then you can access it like you would any other cloud instance, but it's all yours. If you get someone like Cloudflare, you're getting compute space on another server. Same with AWS GCP, you're getting compute space or an instance and you do have an option to get a bare metal instance, but it's still an instance. There's still the hypervisor running behind it. There's still the backend access. There's still a significant kind of surrender of control over that system on most typical cloud providers that you don't see when you have Equinix.

[00:02:36] JM: What are the applications that make the most sense on these different types of cloud providers?

[00:02:42] TB: So I think if you're going to be running applications that have very unpredictable scale that are dependent on large amounts of data, large data sets, typically that's going to be something you're going to want to see on a larger cloud provider. If you have to have a multi-terabyte data store, you can do that on metal, but it's probably going to be impractical because no one wants to manage that. Leveraging managed services for those kinds of things, that makes a lot of sense to put on cloud. If you have things where you have a much more predictable workload, if you have things where you need specific hardware requirements like GPUs or storage requirements that you're not going to be able to customize within available instances at another cloud provider, you're going to want to do those on metal at Equinix. If you have concerns around backend transfer between data centers, leveraging Equinix's fabric is going to help you out as far as speed and cost that typically using a cloud provider may be you can get speed, but certainly the cost is going to be quite the sticker shock in my experience.

[00:03:53] JM: On the major cloud provider side, there are all these managed services. So you have like managed database services and managed container services and managed database services. On the bare metal cloud provider side, you may have more control, but you might lose out from these kinds of managed services. What's the trade-off in managed services versus just total control of the infrastructure?

[00:04:20] TB: I think the biggest trade-off is engineering hours. When you have a managed service, there's a lot of time that you don't have to spend spinning up hardware and the care and feeding thereof. You don't have to do a lot of the things like backups. You don't have to do where you're going to store them or how to create them, testing them. Well, you should still test them, but it's going to look a lot different. You don't have to have the engineering hours dedicated to patching and carrying feeding the same way you would as if you managed it yourself. Most of the appeal of a managed service is that I need to have a database of X-size and I'm just going to put a Terraform command or an API call and now I have it. I don't have to go and provision hardware. I don't have to go and you know load the OS on these things so that now I can run a database software.
It's for ease and simplicity and speed.

That said, if you have very specific use cases that require specific settings, that require specific versions, that require specific hardware configurations, that's going to be something you're not going to have enough control of over in a managed service, right? Managed service I've often said is going to be for like 90% of the use cases out there. Perfectly fine. It works just fine and you don't have any hassles and you can run with that. But if you don't fall within that box, then you start kind of trimming the edges, trimming the corners and pruning your application to fit in a managed service and sometimes you have to make a lot of concessions that you wouldn't need to if you managed your own. But on the other side of that, you still need to spend the engineering hours, you still need to spend the infrastructure hours to set these things up. But on the back side that's probably going to run better every application if you don't fall within a set of use cases where managed services makes more sense.

[00:06:05] JM: You've worked at a variety of different cloud providers yourself. So you've worked at AWS. You've worked at Rackspace. You spent some time at Elastic. Tell me about how you've seen the evolution of cloud over the last six to ten years.

[00:06:25] TB: So it's been interesting to see where cloud was first a curiosity more than anything else. Then became something that you would use for development so that you could spin things up really fast and do some things and then put them back on your data center, in your data center. Then it came to the point where you were essentially using the cloud as a data center. You were decommissioning servers in the data center and then spinning them up as instances in the cloud. And then to see "digital transformation" into cloud where you have managed services and you have serverless and you have containers and you have various APIs and different styles of observability and into what is we see the cloud ecosystem now, which has spawned software development practices around that. You do continuous integration, continuous deployment now. It's really easy because you can just – You commit something to Git and then everything is done automatically and elastically. And that was something that you didn't really see the capability of in data centers or at least not easier. And data center providers, especially in operating almost in a legacy mode, became for a few use cases that were either settled with compliance requirements or that were just too expensive or too hard to move into the cloud, but no one was thinking of innovation in the data center anymore. They were always thinking of innovation in the cloud.

The data centers have caught up so. They've started offering elasticity in the cloud. And I think a lot of the open source tools for container orchestration have helped people move into data centers because now you can just federate a single cluster across a cloud provider or in the data center and it becomes agnostic. Cloud providers themselves have started to jump on the bandwagon offering things like Google Anthos or AWS Outposts where you can run a cloud ecosystem inside an on-premise data center, even air-gapped if you need to, and that's something we saw when I was with the Elastic. The Elastic enterprise, it was one of the first things I saw where it had an entire container orchestration system that was deployed within your data center that acted exactly like the Elastic cloud did to run your stack, and that was pretty impressive. And then you saw MongoDB do the same thing. You saw other SaaS providers doing the same thing. And now the cloud providers are the SaaS providers that are doing that thing to allow you to run the cloud literally anywhere even in a data center.

[00:09:02] JM: What I find amazing about all these businesses, all the different cloud models of businesses that it seems like people rarely shut down infrastructure. It's mostly just people deploy new infrastructure and leave the old infrastructure running. Do you find that generally to be the case? Do people actually shut down infrastructure ever or is it just grow and grow and grow and grow and grow?

[00:09:25] TB: It's rarely the case. I'll be honest. Having worked at cloud providers or cloud optimization MSPs, one of the things we saw when the pandemic hit was that companies that needed to downsize their infrastructure, they couldn't. The ability to scale down was not built into their architecture. They could scale up. Everyone could scale up elastically, but having to find ways to make dataset sizes smaller to reduce the number of redundant instances and high availability configurations just wasn't there. They were meant to replace it easily to be resilient to losing one and replacing it, but certainly not just to decrease the cluster size, to decrease the amount of storage you needed. Sometimes you saw just shrinking a table, it wasn't possible for some of these companies because they had never considered the fact that they might need to do that.

So elasticity has to be able to both stretch and contract, and if you don't architect your application or infrastructure with that in mind, you're going to be set with an infrastructure bill and footprint that can either stay the same or increase, but not decrease, and that bit a lot of companies in the hindquarters when they found themselves in an economic situation where they needed to be very cost conscious. I feel like having seen that in the past, because I'm kind of long in the tooth in the industry, I kind of have always kept that as a consideration whenever designing architectures, but you see a lot of folks now that have learned that lesson and when they go through there they have maybe a very small core minimum size of infrastructure and then everything else becomes a nice to have so that they can shrink down their infrastructure to a very small core of essential services and essential servers, essential – The bare minimum space and not having to worry about what it's going to look like if they have to spin down.

[00:11:15] JM: As I mentioned before, you spent some time at Elastic. There's kind of this discussion around whether or not the individual service providers like Elastic or MongoDB or DataStax can compete with AWS in offering managed services for open source software. What's your experience with that side of things? Do you think that these are viable enough debates that perhaps like the open source projects need to be protected by additional licensing? Or do you think that they're doing just fine?

[00:11:51] TB: So there's two aspects of this, and I'll get to the first by saying that I do think that if AWS wants to run you out of business, they're going to run you out of business. There's nothing you can do to prevent that. They've just got too many resources, right? I do think AWS wants to grab up a majority of the people that use these softwares, especially take like Elastic or MongoDB. Most folks, if they're going to spin stuff up, they're going to do it the easy way. They're going to go, "Well, I'm just going to connect it to, the AWS Elasticsearch service, or the AWS Aurora with MongoDB compatibility," because it's easy and convenient for them. It's a single bill, right?

I do think that the SaaS providers are wise to offer things on their service that you can't get on AWS for those that need it. But, again, like I mentioned before with managed services, that's going to be still a smaller segment of the overall ecosystem, right? Not everybody needs a full expat feature in Elastic. Not everybody needs AI ML. Not everybody needs to have these cluster configurations that are available in Elastic cloud that aren't available in AWS Elasticsearch service.

That said, Elastic probably was never going to be able to get all those customers anyway. So as a business decision, it does make sense for AWS to basically capture the drags. But for customers that have more specific needs, more demanding needs, it does make sense to go with the premium services that are offered by the SaaS providers. The second aspect has everything to do with what you think, the the philosophical connotations behind open source, right? Open source initially wasn't supposed to be really for profit, right? It was things that you built, that the community that helped built, that was for the good and the enrichment of the community. And if you made money off it, it was to the kindness of other people's hearts. Or maybe you built a product based on something like that but it was okay because anybody else could do it.

So if you're a for-profit company and you're trying to license a product so that other people can't use it to make money off of and, only you can make money off it, that's not open source, right? You can't say it's open source and you can no longer try and angle it or spin it to say it's open source because it's not. You have the very tenets of what open source meant when the movement really first came out. Like I said, I'm long in the tooth. I remember when a lot of these kind of movement really started to take flight. It was unthinkable. I remember when Red Hat went IPO and everyone was just shocked that a company would become a for-profit company and then have a such gigantic introduction to the market based on an open source software and everyone saw that and said, "Let's do that too." And that's how you get from 1998 to here we are in 2021 where a group will design something that's open source. They will take the feedback and work of the community to get them to a certain point so they can make profit off of it. The open source you know council consortium, whatever it was, will now become a for-profit company and then still try and say they're open source or have an open source product or have an open source version but they're still rolling those open source contributions from other folks into their for-profit product and then restricting who can actually use it. That's not open source.

[00:15:17] JM: Talk about how infrastructure is managed at Equinix where you work. What kinds of platforms are you guys using to spin up and manage infrastructure deployments?

[00:15:31] TB: So essentially what we have is we have an API that takes you know data that you need around putting up a specific type of server with specific configuration and feeds it into a provisioning system, right? We are doing it and similar to AWS whereas you have a certain types of systems you can spin up. It's not like a McDonald's menu where you can – Well, I shouldn't say McDonald's, but it's like a menu where you can customize some of this and put some of this from there and move that from there. You get this type of server or that type of server or that type of server and those are your options for the most part. And then all we do is we take your API request to say, “I need this type of server with this host name, with this set of IP addresses using this operating system.” Whether you want it behind BGP or some other configuration options like that. And then we provision that server and then we turn that server over to you. And that's the end. We don't put anything else in between. We don't have hypervisor. We don't have an access to anything else like that. That is now your server. Here you go, right? We took what was a process before putting in a ticket, which is what you used to have to do. And even when you had an API, which is an API for a ticket, and we have a completely automated system that does that provisioning with servers that are already racked and stacked.

[00:16:50] JM: And you are a principal solutions architect. So what is a typical client interaction look like for you?

[00:16:58] TB: It's interesting because I don't know that I’ve had a typical one yet. Normally what happens is I’m brought in when the customer has questions about using Equinix in order to make a decision whether they want to purchase or whether they want to expand into using it. Oftentimes they’re network considerations. Oftentimes they're hardware specific considerations. But one thing that I like to do is I like to dig deepest to figure out what the question is. What's the overall question? What's the overall goal? You may be asking a question for a specific network speed or network interface type, but what you really need to know is that, “Hey, will I be able to deploy containers across this in various data centers and will they be able to speak to each other?” And so that's the question we need to find out. And so that's a lot of my interactions is asking the questions to get to what the root cause of what you really want to do and then working backwards from there. So I want to find out what you want your end state to be and we're going to work backwards from there to try and fix those problems or answer those questions beforehand. And that's usually a lot better than going through and saying, “Oh! Answering a question,” and then having to have someone follow up with another question, and then having to have someone follow up with another question. It's much better to talk with people, figure out what they want, figure out what they need and then let's work together to see if we can make that happen.

Most of my interactions with customers, they're technical, but they're conversational. We're not going to bang out specs. I usually don't draw anything on a white board or anything like that. I want to talk to you. I want to figure out what you want to do. We're going to discuss these things. And a lot of times I’m helping them guide themselves on solutions. I’m a sounding board. I’m giving them cautionary things, some best practices things to consider, and then adding details and information so that they can make a good decision about what they want their architecture to be.

[00:18:48] JM: So as a customer becomes larger and has more and more requirements and more operational duties, do you give them like additional services? Do you help them install or provisional additional services? Like do you help them figure out how to build scalable monitoring and metrics and logging systems or security systems? Tell me more about your role as a company scales.

[00:19:15] TB: So as a company scales, I involved with kind of at the beginning saying we want to order this, but we also want to eventually move to this part, and we're going to have those discussions early about saying, “Okay. Well, if this is what you want to go to, these are the considerations you're going to have.” We will talk about observability. We will talk about logging. We will talk about resilience, high-availability, disaster recovery, SLOs, SLAs. All these things have to be discussed because they have to be built into the architecture. And then once you know what it looks like, once you know what that architecture's going to look like, then you're going to know what you're going to have to buy, right? Because you can't just go in blindly with these and make decisions. That said, we don't build it. We don't do professional services. I will mock up a POC. I’ll demonstrate some things, some of the capabilities especially around a specific use case. I’ll throw up a proof of concept and then let the customer test drive it. And I can talk them through building some of those things, but we don't actually build them. They will have the resources to do that. But, again, I’m going to give them guidance. I’m going to give them some best practices and things to consider. Warn them about some pitfalls and kind of advise them on what I think would work best. But in the end that's still the customer's responsibility.

[00:20:27] JM: Can you give more of a description for how a bare metal as a service cloud compares to a public cloud in terms of latency and infrastructure and just what goes on in a round trip?

[00:20:41] TB: Sure. In essence, whenever you have any kind of hypervisor running as an abstraction between the user and the end result, you're going to have latency of some kind especially as it interacts with more and more items. For example, if you have a standard kind of AWS instance that has EBS block storage and logs everything to CloudWatch, you have several layers of APIs you go through for everything. You have the network API that you have to go through that's going to route virtually things virtually. You have your buckets for both bandwidth usage. You have buckets for CPU usage. You have buckets for memory allocation. You have buckets for I/O and all these things, all these bean counters exist. You have throttling on any of these API calls, internal APIs, external APIs, service limits. You have to go through, and all these things are checked constantly. You do run storage to the network. Some things do have local disk, but those are the rare ones. Any instance that is not a bare metal instance still is abstracted through a hypervisor even if that instance is on a specific type of hardware that has access to like NVME storage or SSDs, right? You're getting a VM inside a VM inside a VM that has any large number of beam counters and APIs that it interacts with to do everything. It’s like you're wearing VR goggles that you can interact with the world, but certainly through the lens and protections, gates and guardrails that are set up for you. Bare metal is not that. You have the server. It's not network abstracted. You actually have the NIC cards that are physically connected to the medium. It's all yours. The SSDs, the spinning disks or the NVME storage, that's all you. You're not having any abstraction layer. It is you in the operating system and the metal as if it was a laptop sitting on your desk, a server in the back office. You don't have any virtualization in between there to start off. Now, you can run VMware on it if you want to. You can install whatever you want to on there after that point. I mean, if you want to have that abstraction layer so you can run Kubernetes containers, or my favorite, running a Java Virtual Machine inside a container inside a VMware virtual machine, a VM inside a VM inside a VM, you can do that, but that's not what you're going to get by default. You're going to get you. You're going to get your root keys and you're going to get a metal server.

[00:23:21] JM: So across all these different workloads, you described like containers and VMs and JVMs and different configurations, are there any like infrastructure-wide problems that can occur that they're so difficult to debug that the cloud provider has to get involved? Like that you as a solutions architect would have to get involved in a debugging process?

[00:23:46] TB: So in a public cloud, quite often. Very often as a matter of fact, because you have API service limits, because you have CPU buckets, because you have I/O buckets. These things are not known. They're not transparent to the customer. You have to actually have – Like an AWS, you actually have to call support be able to go through some logs that go through some internal systems to find out if you've been banging off of API limits. Or you have to log it externally to CloudWatch, and then for a very you know marginal, or sometimes not marginal fee, you have the privilege of being able to go look at these logs or put them into S3 and then run something else against them so you can do monitoring because you don't actually have access to the logs that are telling you that you're hitting API calls or API service limits.

It can be – If those service limits can be raised, then you have to go have another interaction to have them raised so that you can try and run your application or your workload without hanging off these limits. And the number of hoops that people jump through to circumvent the limits, which are completely arbitrary, it's staggering. When folks will split up their workloads between 10, 15, 20, hundreds of accounts just to make sure that they are adjusting for API and other service limits, it becomes a little hard to manage. Literally, you're now managing the accounts and the payments on all those accounts and the service limits behind all those accounts in addition to the infrastructure management that you have to do. It becomes harrowing. You may not bloat your code, but you're certainly going to have to bloat your engineering hours, your administrative hours and your finance hours to keep track of all this, right?

It's why companies like CloudHealth exist, specifically because of all these things that you have to keep track of and to get observability into a system that you wouldn't have otherwise because you have to aggregate all these things from various sources that you – And even then, you still don't have access everything. You look at putting everything into the cost and usage reports to figure out what cost and spending is, because I have all these various buckets and all these various usage types. It's a lot. It is a lot to deal with. And so you're looking at that and then you have the other side, is like you've got with Equinix Metal. You've got CPU time and you've got network. That's it. You charge by the hour for your instance type and you get charged by gigabyte for network, and that kind of simplifies things quite a bit versus everything else you have to do. You don't have to – Again, for me as solutions architecture, I typically don't get involved once a person has made their agreement or once an org has made their agreement to use Equinix Metal. Typically I don't get involved. We do have TAMs that get involved for ongoing operational support. And they will, if someone runs into problems on the platform, or has concerns about how to use it going forward, or has concerns about, "Hey, I need to perform maintenance or I need to deploy these things," the TAMs will absolutely get involved in the system with that, but there aren't limits that you have to deal with. There aren't the number of various concerns of how the strings are pulled that you have to deal with because you still just have bare metal server on a rack with some network and power connected to it. And that reduction in the set of concerns you have while running your applications can be huge for a business both in a cost savings aspect, in an agility aspect, in a management aspect for the internal practices, for compliance reasons, like it becomes a game changer when you can have the kind of application elasticity you need without all the overhead that goes behind it.

[00:27:27] JM: Tell me more about what you as solutions architect need to know about the hardware itself or the infrastructure deployment itself. Like do you need to understand the hardware at a deep level?

[00:27:41] TB: I don't know if it has to be like super deep. Like I don't necessarily have to know the intricacies of the CPU. I don't necessarily need to know the die size or how many transistors are on the CPU, right? But I do need to know the hardware well enough to figure out what's a good use case for it. I've had customers recently, we're speccing at a server that looks perfectly fine for them, but they said, "Well, I understand that this server has this number of cores available, which is great, but the application I have can only pin to one core at a time so it can't multi-thread.” And it also is clock speed dependent. So while this has this 32 cores at 2.2 gigahertz sounds great, but I’ll only get 2.2 gigahertz. And what I really need is something that's three gigahertz or higher because I can only use one core, right? And so you have to know that. You have to know whether that piece of hardware is going to be the right fit for their use case. If they need something that's stateful and needs high speed storage, right? You have to have a good idea of what the I/O bandwidth and what the transactions per second are, or writes per second, or reads per second are on the storage. If SSD is going to be fine, or if the NVME store is going to be better for you, what the size of the storage is? What that footprint is? Dow do you need a lot of storage? Do you need to have some boxes for CPU that are stateless? Do you need to have some boxes that are stateful so you need a lot of storage? Wow what does your overall architecture look like and how many of which do you need? And so those are considerations. We do have to know the hardware well enough. We need to know what their network usages are, like what their network concerns are to know whether we need to have something that has four cards or two by 25. These are all things that we have to consider. So we have to know the hardware and speak the harder well enough to know what it's capable of by default and certainly what can be customized. If we're not super familiar with it, I have to know how many GPUs we can put in something. If it has room to throw another NVME card in there. If we can put more RAM in it. What type of CPUs can we swap out if we need to swap out CPUs? Those are things that are definitely great for us to know. So you have to have a fairly extensive knowledge, but you don't have to be – I don't think I have to be an expert. I don't have to know what the chipset is on the north bridge or anything like that.

[00:30:02] JM: Are there any particular DevOps tools or infrastructure tools that you've seen really gain popularity in recent years that you've seen companies gain or save time and money?

[00:30:15] TB: I think it's an interesting question you asked, because DevOps is the practice. It's the culture. It’s ideology. And there are tools that help that, but I feel like you've got automation tools, right? Automated deployment, or container orchestration, or configuration tools, right? All of these combined together help the DevOps culture, help the DevOps practices in an organization certainly, but I do think if you're talking about like infrastructure, right? I think Terraform has been a game changer. Ansible I thought was certainly going to have taken over the – if you'd asked me five years ago, I thought Ansible would have taken over the industry. But I feel like Terraform is the one that's really become much more ubiquitous. AWS would rather use CloudFormation. And we're being honest, CloudFormation is not great. It has come a long way over a few years because it was really, really not great, but I don't think it's ever going to to really meet where Terraform has because of Terraform's platform agnosticity, right?

I think one of the great things that helps us at Equinix Metal is having our own Terraform providers so you can just have Terraform like treat Equinix Metal like another provider in your infrastructure so you can provision here, you can provision there. You write your modules and run them. But I think another set of tools that has really helped out is the entire ecosystem around container deployment, Kubernetes and all those helper applications for configuration for learning and monitoring. So you talk about running the stack. You use Terraform to provision your underlying hardware. You use Kubernetes to deploy and manage the containers that run on that hardware and then you would run something like – Whether it's Jenkins, whether it's TeamCity or any other kind of automation tool that when you deploy something to Git – And me not forget to mention Git and GitHub or/and GitLabs and the other Git paste software version control tools and deployment tools. But when you deploy something or commit something to Git and once you push it to master, that it triggers all these things that that now make your changes live on the internet.

So the evolution of those things has been gradual and certainly with bumps and things that we thought were going to be the winners and aren't winners, right? The things that really come out on top, the ones we see now like I mentioned, Terraform, Kubernetes and Git, GitHub. Whereas, like I said, five years ago, maybe it would have been like, "Oh! It's going to be Ansible and it's going to be Mesos and then it's going to be –" Five years ago probably sort of been Git. But if you look at things like how do you do secret storage? How do you manage just your projects and your sprints and things like that? And so a lot of these tools helped us do automation of the manual processes. Automation of the manual processes that go behind a DevOps culture, but I think we also need to take the time to speak the other tools that actually help people do the actual development and software development sprints and all that kind of stuff like that, because that's what the real DevOps is, right? Your agile planning, your standups, your Kanban boards, your waffle boards or whatever it is that you use, right? The observability and the feedback that you get from operations or site reliability engineering, the response to incidents, RCAs, and then what do you do next? And now we need to integrate that into our sprints. Like all those practices, all those aspects of the culture, to me that's the DevOps tool, right? That's the thing that you use to really make the difference. And then the other thing is, like I said, they're just automating the things that you would have to do to deploy.

[00:33:45] JM: Well, as we begin to draw to a close, I’d just love to get your perspective on where cloud infrastructure is going. What's changing and what you see as predictions in the near future?

[00:33:57] TB: So I think we went from a few years ago where we were just really starting to break out of instances into containers and then on into serverless, right? People had been doing microservices well long enough ago, but the necessary compute has gotten smaller and smaller and smaller, right? At this point I think where we're going to look to is the expansion far more into the edge and what the edge looks like between 5G, between K3S and other types of workload and container management on edge compute and getting high speed data transfer almost within yards or fractions of a mile from your end customer and have double digit millisecond transmission from my desk to a data center. That's going to make a huge difference. The edge is really the new frontier and how much we can get at the edge and how configurable it's going to be, how secure it's going to be. As homes and businesses become more and more entrenched with IoT, all these things are going to need to phone home. They're all going to have to have varying levels of compute on the device and varying levels of compute that they have to have at the edge so they can deliver good service and then they have to be done securely, right? There's a huge opportunity and a huge vulnerability for computing at the edge. And the security thereof is going to be I think one of the biggest things that we need to manage. And what that looks like, I don't know yet, but we're going to need to figure that out and we're going to figure that out pretty quickly, because as innovation rolls along, the edge is going to be the new data centers. I feel like data centers are going to be for aggregation and moving things from the other, but most of that stuff is going to be done within yards or miles of the end user.

[00:35:47] JM: Okay. Great. Well, anything else you'd like to add, Tim?

[00:35:51] TB: I just want to add that you know I think everyone should kind of do what they can to give everyone else a hand up here. We're living in some really crazy times between natural disasters and just society as a whole. So take some time to help someone, whether it's helping at your job, helping the neighborhood. Give time and space for people who need it and give time and space for yourselves.

[00:36:16] JM: Well, thanks, Tim. I appreciate you coming to the show.

[00:36:19] TB: My pleasure, Jeffrey. Thank you.

[END]