EPISODE 640

[INTRODUCTION]

[0:00:00.3] JM: Edge computing is a term used to define computation that takes place in an environment outside of a data center. Edge computing is a broad term. Your smartphone is an edge device. Your self-driving car is an edge device. A security camera with a computer chip is an edge device. These edge devices have existed for a long time now, but the term edge computing has only started being used more recently. Why is that? It's mostly because the volume of data produced by edge devices and the type of computation that we want from edge devices is changing.

We want to develop large sensor networks to enable smart factories and smart agriculture fields. We want our smartphones to have machine learning models that get updated as frequently as possible. We want to use self-driving cars and drones and smart refrigerators to develop elaborate mesh networks and perhaps even have micro payments between machines, so that computation can be offloaded from edge devices to a nearby mesh network for a small price.

Kubernetes is a tool for orchestrating distributed containerized computation. Just as Kubernetes is being widely used for data center infrastructure, it can also be used to orchestrate computation among nodes on-premise at a factory, or in a smart agriculture environment, or in a mesh network.

In today's episode, Venkat Yalla from Microsoft joins the show to talk about Kubernetes at the edge and how Internet of Things applications can use Kubernetes for their deployments today, and also what the future might hold.

Full disclosure, Microsoft where Venkat works is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:01:50.6] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques. DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW]

[0:04:14.1] JM: Venkat Yalla, you are a senior product manager at Microsoft. Welcome to Software Engineering Daily.

[0:04:18.9] VY: Thanks, Jeff. Good to be here.

[0:04:20.4] JM: We've done a few shows recently about edge computing. We've talked about sensor networks in an agricultural field, we've talked about machine learning at an oil refinery, we've talked about a cruise ship that operates its own data center. I would classify each of these areas as edge computing. You have lots of computation in these environments that have historically not had very many computers involved. How do you define edge computing?

[0:04:56.0] VY: Sure. Yeah, so all the examples that you gave are definitely good examples of edge computing. The way I like to look at it is really coming down to two key phrases and that is data gravity. Wherever you have lots of data being generated, automatically that actually attracts compute to the data, right? When you talk about the edge, especially when you think about things like oil refineries, or think about for example a connected stadium which has cameras that is looking at audience to make sure everything is going fine, that's generating huge amount of data.

When you have that much amount of data, you want to apply some compute to it, but the concept of data gravity tells us that the motor data is so much that it actually attracts compute to it, right? In all of these scenarios that you talked about and the scenarios that we are seeing in the future and also the present, is this data gravity at the edge, and that is attracting compute to it. When compute is attracted to that data, and that's what I would call edge compute.

[0:06:00.1] JM: That data gravity, that idea, I want to understand the economics behind that, because people have come on the show and they've said this idea that it's oftentimes cheaper to bring compute to the data as opposed to moving the data to a place where the compute is. Why is that? Can you explain that in more detail?

[0:06:21.0] VY: One of the primary reasons is of course the economics of it in a sense that is just amount of data that there is, right? Let’s take a canonical example of smart camera. That is nowadays, most of these cameras are 1080p, some are even 4k, so its generating a huge amount of data every second. There are two options here, you either take all of these video frames and ship them all to the cloud and you have the cost of ingesting the data, the bandwidth cost, etc., and then you can apply the vast compute available in the cloud.

Really, what you're really looking for is that one key frame, one key frame if it's a security camera where someone is – where it has detected a human where it shouldn't be, right? That is the key frame that we're really looking for. The rest of the 99% of data is not interesting. With the concept of data gravity, if you can bring compute to the data and now have a closed loop where this data is being evaluated at the edge, and then when that key frame happens, right, where the human is detected, that key frame is then sent to the cloud.

It's a lot more cheaper to actually get the insights you want. Once that key frame is detected, you send it to the cloud, that can set off a whole bunch of actions using – could be in a whole bunch of technologies like serverless, or whatever and be able to send that data to act on that key information and send alerts if required. If you look at where the supply is economically, it makes it really – it makes a lot of sense to apply compute closer to the data.

[0:07:55.6] JM: Right, because the cost of sending the code to run a MapReduce like operation over all of the frames of a camera that's sitting on an oil rig, that's going to be a lot cheaper than the cost of sending all of those frames from the camera to the data center, where that MapReduce operation might be processed.

[0:08:22.6] VY: Exactly, exactly. Then you bring up another good point when you talk about the oil rig, where you probably don't have a stable connection to the cloud in the first place, right? You're in the middle of nowhere in the ocean, you probably have very expensive satellite connection that you turn on maybe a few times a day, but you have all of this data that's being generated every second. Are you going to then process data only when you have the connection and that turns out to be a lot more expensive, or being able to process a lot of this data offline as well, right? That's a key scenario where edge compute plays a role.

[0:08:55.6] JM: I guess, this raises the question, who exactly is it's going to be detected on the security camera on an oil rig? Are these oil pirates?

[0:09:06.7] VY: That's only a possibility. Yes. For the oil rig, it's more about obviously, the drilling head to whatever that's going down into the ocean, right? Yes, definitely. I guess pirates could also be – it's not unfathomable.

[0:09:21.6] JM: Okay. There's also a trade-off here with the machine learning stuff, right? Because with machine learning, you have advantages that you can get from custom hardware, and then so you have this interesting trade-off where it's like, do we give custom – do we put custom hardware on the oil rig now, because then that would mean we're starting to move stuff, move hardware from the data center to the edge, which starts to sound troublesome, because then it's like, do we build the entire –

[0:09:53.0] VY: I wouldn't really call it troublesome. In the end, what you're processing at the edge is very scenario-specific. For example, in the oil rig example, or in the smart camera example, it does make sense to have some kind of acceleration for ML workloads if you are running those models on the edge. What you can do is you can have a server class, or a desktop class system with a desktop GPU, or an FPGA, because you know that it is a scenario-specific thing, and that's the good part about the edge.

With the cloud, you almost have to – you have the least common denominator in the sense that you have to have a generic compute, generic GPUs that are – because that is what the – because you want to be able to apply different kinds of workloads on to that compute. When you talk about the edge, it's very specific to your scenario and what your cost structure is and how much you're willing to spend on the hardware and what are your requirements around latency, what are your requirements around performance, and you can customize the hardware that resides on the edge to those requirements, right?

I wouldn't call it troublesome, but definitely the ecosystem is quite broad, so you might have something from a Raspberry Pi to a server class system with an FPGA running on it. It's a pretty broad space, I would say.

[0:11:07.8] JM: Right, okay. We have a broad space. We have the security-camera example, we have the oil rig example, we have these examples such as the agricultural field, where you might have a lot of sensors in an agricultural field. In that scenario, maybe you actually have cellular connection so the connection might be a little bit better. In any case, there's a lot of different axes we're talking about here, like connectivity, the number of devices, the type of device, the type of processor. You are on the – I believe the IoT edge team in Microsoft. You're trying to build software that is able to encompass all of these use cases, or at least some software that encompasses all of these use cases, and then perhaps more domain-specific software, or make more domain-specific patterns for other areas within that broad domain of edge computing. Am I mapping your job title to what you do accurately?

[0:12:09.8] VY: I mean, absolutely. You got it. Yes, I am in the IoT edge platform team, and we are building a software platform that can scale across all of these uses, right? One use case that we didn't really touch upon, but is really important is a use case where you cannot afford the round-trip latency to the cloud. Think of for example a self-driving car, or a speedboat, right? A speedboat is good, because you're offline because you're again in the ocean somewhere, and you want to be able to detect certain problems in the engine, cut it off, make sure it doesn't blow up and all of this round-trip time you want to be able to do it locally.

You cannot afford – even if you had a connection, you cannot afford tens of milliseconds of latency that is required to actually process this data. Now you want really sub-millisecond, near real-time, and that's where again edge plays a big role. Yes, you're absolutely right.

[0:13:02.0] JM: I find all these different areas so interesting. From the self-driving car computing example, to the oil rig example, to the camera example, what are the commonalities across these different domains? When you're trying to define what you want to put in an edge computing platform, what are the commonalities that you will build into that platform?

[0:13:26.5] VY: Yeah, great question. One thing that we like to think about and one of our design goals is to have a consistent user experience from developing the software that runs on the edge perspective, as well as for managing edge software, because when we talk about edge and IOT, the scale is a lot higher than traditional platforms. Let's tackle the first one in the beginning, right? This is the developer experience. What we want to do is leverage the same cloud development scales that you would use for your device, or your service that lives in the cloud, and be able to map it with very little changes to the edge.

You don't have to now say, “Okay, here's my cloud solution. I need to now go to the edge, scratch that, let's start from the beginning here, right?” That is something that we don't want to do. The bet we made and what really helps us do that is the bet on containers, or OS level containers, where you can take your application, or your service, break it down into microservices, containerize them, which is best practices nowadays even in the cloud, and now use those same containerized workloads and move them to the edge and have perhaps not exactly the same amount of processing power, but very minimal changes be able to run very similar code on the edge as well. That is something that is common across all of these domains and something that we have worked hard to actually bring to our platform.

[0:14:59.0] JM: We've done a lot of shows about containerization and container orchestration and why containers and Kubernetes specifically is a useful medium for managing clusters, for managing resources. Why is Kubernetes a useful tool for running software at the edge?

[0:15:19.5] VY: Yeah. Kubernetes itself is for example, we don't have a strong dependency on Kubernetes per se. If you look at the software on the edge, we have something called the IoT edge runtime. Part of the IOT edge runtime is a Kubernetes-like component, which itself is containerized, but that is – the job of that component, the edge agent is to maintain desired state on the system.

The way you would start off is you would go into your console in the cloud, define just like you would from a Kubernetes manifest, but this is an IOT edge manifest where you would define that these are the various modules or the workloads that I want running on my edge device and these are the patterns, or the paths of communication. In the end, once you've defined this pipeline of compute that really the end product is a JSON document. That document is then just shipped to the IoT edge device where the edge runtime then parses it and pretty much makes it so, right?

Now where Kubernetes comes in is because we are using open technologies like containers, we can define this edge manifest that I talked about in the form of a Kubernetes deployment. Now you can leverage your Kubernetes skills and all of the niceties of that platform, like managing your secrets, doing at scale deployments, all of that, and map it to your edge device. We can go into the details more on how we do that.

[0:16:50.1] JM: Do you have an example? Is there a prototypical example of a company, or a type of deployment and edge computing deployment where the model of containerization has proven to be useful?

[0:17:04.2] VY: Yeah. In almost all of the applications we have, containerization definitely has proven to be extremely useful. For example, let's talk about canonical case of doing an ML workload, correct? Over here, we know that the workloads on the edge are not static. For example, you have your – let's go back to our camera example, and in that you have a model that you have trained in the cloud, but are executing on the edge. It is able to detect a person with 80% accuracy. As you run this model, you get more and more data that you can actually now train in the cloud again and build a model that has, let's say 90% accuracy. Now you want to update your edge devices with this new model.

Because we are based on containers, what you would do is you would package this into using an Azure ML, for example, container and package this new and improved model, and all you have to do is change one line of coding of deployment to go from version one to version two. Immediately, the edge agent which is part of our platform detects this, takes down the container that was running and brings up the new container. That flexibility is really useful. We see that customers love that almost more than anything else in our platform, the way to manage your devices on your terms and easily, right? That's definitely where containers come into the picture.

The other example, I would say is you might not want to write all the code that you want to run on the edge, right? Just part of it. An example is what we allowed you to do in our platform is not just have one container, but multiple containers. You don't need to be responsible for every container. Let's say you go to the Azure marketplace and realize that, "Hey, here's this machine learning model from Azure ML that I can use in my edge solution, so I have my business logic that is running in one module, that is extracting data from sensors and now I want to feed it in to this other container that is pre-built for me, that I can just bring out to my edge device."

That can now go to another Azure function that acts up when there is some insight that is gleaned from the data. You want to interleave custom code that you build with code that you have from third parties and manage it all in a microservices fashion, so that you can update these things independently. There again, the paradigm of containers really comes into effect and help some.

[SPONSOR MESSAGE]

[0:19:39.0] JM: Sponsoring today's podcast is Datadog, a cloud scale monitoring and analytics platform. Datadog integrates with more than 200 technologies, including Cloud Foundry, Kubernetes, Docker and Kafka, so you can get deep visibility into every layer of your applications and infrastructure in the cloud, on-premises, in containers, or wherever they run.

With rich dashboards, machine learning-powered alerts and distributed request tracing, Datadog helps teams resolve issues quickly and release new features faster. Start monitoring your dynamic cloud infrastructure today with a 14-day trial. Listeners of this podcast will also get a free t-shirt for trying Datadog. Go to softwareengineeringdaily.com/datadog to get that free t-shirt and that 14-day free trial. That's softwareengineeringdaily.com/datadog

[INTERVIEW CONTINUED]

[0:20:42.7] JM: The machine learning stuff is obviously interesting. What about a more basic example? Let's say there's a big candy company, a candy company comes to you and says, "We've got our candy factories. They've got computers in them. They've we've got conveyor belt computers, maybe we've got some security camera computers, some of it is old software, some of it is new software. Help us modernize our factory." Where would you begin with that? Because this is an edge computing domain, you have a bunch of connected things in a factory. How do you get them on a platform like IoT edge, or if you want to talk about Kubernetes, what is the software stack that the candy factory should adopt?

[0:21:32.8] VY: Yes. At the baseline, they would probably – the first thing they need is to select an edge device, right? They would select the device that is conducive to their use case, their price points, etc. On that edge device where we would start is for running IoT edge, you have to take a bet on containers.

[0:21:53.0] JM: By the way, this would be a server that they would actually just put in their building, on the factory floor, or whatever?

[0:21:59.7] VY: Exactly. Yeah, this is just a server desktop class, whatever. It can even be a Raspberry Pi, depending on what their scenario is. Yeah, it can span across all of this hardware, right? Then they would talk about, okay, is this a green field, or a brown field as we call it, right? Green field is where they have the luxury of starting from scratch, starting from a modern framework using containers, Kubernetes, IoT edge or not, right? Brown field is where they have existing software already that they need to containerize.

Again, because we take a big bet on containerization, what we would do is try to figure out in their old stack what they can containerize. Usually if it is a Linux-based system, it almost always is possible to containerize their legacy code base and run it as part of a container. Then they would integrate their legacy code with some of the SDKs that we provide as part of Azure IoT. Now these SDKs are available in multiple languages, so in most cases they have a language that they can use and integrate the data that they produce out of their legacy software using their SDKs and to now talk to the new architecture.

Once they have this new architecture, they are off to the races where they can either send that data directly to the cloud, do some pre-processing on the edge using other custom-built modules, or modules that they have bought off our marketplace and pretty much be able to modernize their entire solution. That would be a pattern that we would recommend for a brownfield solution.

[0:23:33.1] JM: These brownfield situations, I don't know a whole lot about enterprise software for factories, but I imagine some of it is proprietary binaries that were not built by the candy factory. Are these kinds of applications, things that can be containerized, or are there just certain parts of their build that they're going to have to leave outside of a container?

[0:23:59.5] VY: It depends again on the scenario. Yes, depending on how there are – the legacy software is built, for example if it's a WinForms application that has some GUI dependencies, right, which cannot easily be containerized. Then in those scenarios, it would be more difficult to do something like this, where they containerize the pretty much lift and shift their all solution and move to the new one. What you would really do is then take a phased approach, where you start moving various parts that do – that can be containerized, that can be modernized somehow building translators to translate between their legacy stuff and new stuff, with an overall goal of moving everything over.

Then again, take a phased approach where you have some pieces that are running on legacy hardware for example, one edge computer or an edge device could be running old hardware with the old software stack, and there is some translation that happens to an edge device that is now more modern.

[0:24:59.6] JM: Yeah. Or they could build some shim, or some another interface to this software that's hard to interface with, or maybe it's a read-only interface into that other piece of legacy software. In any case, they get – I think the advantage of going through this installation is they now have a platform that they can deploy new stuff onto and in a very controlled fashion, and a forward compatible fashion.

[0:25:28.2] VY: Exactly.

[0:25:29.1] JM: What about security requirement? A lot of these edge environments have – since they're interfacing with the real world, like interfacing with your oil rig, or your candy factory, these can have very serious physical impacts on the world. What does it do from a security perspective? Does that change the network architecture?

[0:25:51.4] VY: Yeah, it does. Not really network architecture, I would say, it even changes the changes the platform architecture. Where for example, if you think about a secure data center, right? There you can rely on that particular server to be in a secure location that is guarded 24/7, and it's not really difficult to get physical access to. On an edge device, it could be a small device that is sitting on the factory floor, someone could just disconnect it, put it in their bag and walk away.

The platform that we have built again is flexible to go across the entire spectrum of requirements around security, as part of the edge architecture, or the IoT edge platform, we have something called the edge security daemon, which is a native component and that provides abstractions over various hardware security modules we have. We try to provide flexibility to for various kinds of deployments. We have something called a standard promise, and then have a more secure promise and an ultimate one, where you have more hardware-based security also.

Again, based on your requirements, you can have for example some of your secure data that is stored not on the disk, but you store it in hardware security module, like a trusted platform module, like TPM where you store the keys to your IoT hub and things like that there. That's where our edge security daemon can now interact with a platform-specific library, that they can either get from third parties, or build themselves, that actually works in a more secure fashion.

If you want even more security, then we have something called the arm trust zones, or Intel SGX, which is the secure enclave process. Where not just data is in a hardware-based secure environment, but really even the compute for example, some of the modules, which has very custom logic is now running in an armed trust zone, or an Intel SGX Enclave, where even if someone gets physical access to the system, they will not be able to reverse-engineer, or get access to your secrets.

Depending on how important and how secure you want to make your solution, we have something called ties, right? This is part of the edge certified hardware that is part of the Azure catalog, where we have L1 to L5 tiers. Where you have, if you want basic security, where you don't want cost as your primary factor, scale is your primary factor, you can go to L1 devices, but they will also be in the catalog, something devices that are categorized as L4 or L5, which have much higher levels of security using specific hardware features and hardware security modules. If your scenario requires that, you can pick devices from there.

[0:28:44.8] JM: What I think is really interesting about this whole area of edge computing and the overlap between when we started to talk about edge computing with IoT devices and self-driving cars and all these different devices, what's really exciting is you can start to imagine a world where we are deploying, or a lot of people are deploying software to robotic systems, or physical systems, or software that is heavily integrated with our physical environment. I'm sure there are plenty of people out there listening who already deploy software to these kinds of systems, but I'm sure engaging with this stuff and working on the team that you do, you're getting a preview of how this world is going to look going forward as we do increasingly engage with the physical world with our software, as opposed to just information systems.

I do think about it in the context of even just like the candy factory. We talked about Kubernetes deploying to the candy factory. You could imagine a world where maybe if you're running a candy factory, you have four different candy factories and you could have a Kubernetes cluster on each of those candy factories, you also have a cloud – a Kubernetes cluster in the cloud that's maybe doing machine learning training, or it's synthesizing data from the different candy factories, but in case, you can do stuff like AB testing one candy factory versus another and running controlled experiments between the two candy factories.

We wanted to take the candy factory deployment example a little bit further, and we wanted to set up this world where we've got consistent deployments across our candy factory instances. Meaning, each candy factory has consistent, mutable infrastructure that we can change easily from our console, and then we also have this deployment where we've got a cloud fabric of the candy factory. I don't want to throw too much at you at once, but how would you architect that kind of project for the candy factory?

[0:31:00.1] VY: Sure. Yeah. At first, what you would do is define the devices that live in your company, right? Let's say you’re candy company A, so you have let's say a hundred candy machines that are across 10 sites and each one has 10 just to keep things simple. The first thing you would do is all of these devices would have an identity in the cloud. Now each of these hundred devices would start off with something called a device twin. That means, their digital instantiation of the physical device that lives in the cloud.

You would use very straightforward concepts, like tagging, right? You can now tag each of the devices that exist with a tag to say, this belongs to factory one, this belongs to factory two and you can make it as fine-grained or coarse-grained as you require. Now you have a logical grouping of your devices, so the way it works is you put up a device in a new device in one of your candy factories, it will then reach out to the cloud depending on – it's pretty much an empty box at that point. It will reach out to the cloud to say, “Hey, I have booted. I was asked to come here,” because that's what it was configured in the factory. Immediately the cloud says, “Okay, based on your credentials that you have provided to me that you have securely stored in a TPM, for example."

These are the tags from the digital twin, this is who you are, this is your digital identity. By the way, here is a software deployment that has been configured to be pushed to you, so that now you can configure yourself with the right modules, with the right workloads that make sense for your factory. Now what you have is you have a device on the edge that is connected to the cloud. It has a digital identity as well. What you would do then is use these concept of deployment. It could be Kubernetes deployment, or just IoT hub deployments if you're talking about Azure IoT, HR product. Where you can then say, here's this new deployment and I want version 2 of my software to be running on this deployment. I don't want to push this out to all of my hundred devices.

Like you said, I want to do some AB testing, right? I just want one device in each factory, or just one factory to get this particular deployment, the software deployment. When you create a software deployment, you can actually target your devices to which this particular deployment applies to. As soon as the device connects, it can now say, "You have an updated deployment." The new modules and the new connection comes in, you have you have things updated on the edge device. Does that make sense?

[0:33:37.1] JM: It does make sense. Yes. I think, I want to put a finer point on the delineation between what you are building and when we're just talking about – when we're talking about IoT deployments more broadly. You're on the IoT edge team. We've done a show about IoT edge, we've also going to show about IoT hub. Can you talk a little bit more about what you're building specifically?

[0:34:01.7] VY: Sure. We are the IoT edge platform team. We have an edge component and the edge piece of things, and also a service and the cloud piece of the solution, right? You talked about the IoT hub. For us, the IoT hub is the cloud aspect, or the service that the edge devices connect to. On the edge itself, there needs to be a platform, there needs to be an API or runtime that makes communication to our service a lot more easier, to either provision this service using our device provisioning service when they first come online, or to manage devices with various software configurations.

Also for example, do things like device management where you want to update the firmware of the device, you want to update the OS itself. All of this, you can do remotely from the service in the cloud and that is IoT hub.

You also want compute, or local logic and workloads to be running on the device. That is where our IoT edge, SDKs and the IoT edge modules come into play. They can now, using various tooling that we have built using VS code and visual studio, you can build modules that are compliant with the edge runtime that is running on the edge device. Once you use these SDKs, or these tools, it makes it super easy to take your OT, or the operational technology networks where you have things like MQTT, AMQP and these various IoT protocols, translate them into a cloud native format in one of your protocol translation modules, and use our runtime to then send that data up to the cloud in the IoT hub, where once you reach the cloud, pretty much the world is your domain, because you have a whole bunch of compute, a whole bunch of technologies you can use to be able to work on that data.

That's the delineation where you have modules and SDKs and the runtime on the edge. You have device – management device identity in the IoT hub, and that's a gateway into the entire cloud architecture itself using – could be our databases, could be ML, could be a whole bunch of cloud services that are outside the IoT hub.

[0:36:18.7] JM: Right. If I understand correctly, the IoT hub is where it's like the gateway of communication between your edge deployment and your cloud deployment. That's useful both because you can do all this device management in this centralized place, but also security for example. It's very useful to have a single device that is the secure point of entry of ingress and egress. Is that an accurate description of the hub?

[0:36:51.1] VY: Absolutely. Yes. For example, on the edge device, you have these various modules running, but what if you have a module that is running that you didn't authorize to run, right? You want to make sure that the software configuration that is running on the devices is absolutely the configuration that you were intending to, right?

Using the IOT edge runtime and which works in concert with the IoT hub, it creates these module identities, workload identities and ensures that what is running on the edge device itself is something that is authorized to run. A great point about security there as well. IoT hub with IoT edge runtime make sure of those security promises.

[SPONSOR MESSAGE]

[0:37:40.7] JM: We are all looking for a dream job. Thanks to the internet, it's gotten easier to get matched up with an ideal job. Vettery is an online hiring marketplace that connects highly qualified jobseekers with inspiring companies. Once you have been vetted and accepted to Vettery, companies reach out directly to you, because they know you are a high-quality candidate. The Vettery matching algorithm shows off your profile to hiring managers looking for someone with your skills, your experience and your preferences. Because you've been vetted and you're a highly qualified candidate, you should be able to find something that suits your preferences.

To check out Vettery and apply, go to vettery.com/sedaily for more information. Vettery is completely free for job seekers. There's 4,000 growing companies, from startups to large corporations that have partnered with Vettery and will have a direct connection to access your profile. There are fulltime jobs, contract roles, remote job listings with a variety of technical roles in all industries, and you can sign up on vettery.com/sedaily and get a $500 bonus if you accept a job through Vettery.

Get started on your new career path today. Get connected to a network of 4,000 companies and get vetted and accepted to Vettery by going to vettery.com/sedaily. Thank you to Vettery for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:39:24.9] JM: The IOT edge, to give a finer point on that, this includes SDKs and the runtime for the devices that are in your edge deployment. What are some examples of SDKs? What are some things that you would be running within the IOT edge platform?

[0:39:45.2] VY: You would have for example the workloads, which is your custom business logic, but you would also have other aspects of the platform that the SDK helps you with. One is to get to that gateway, right? You have abstractions to create for example, messages which is – our telemetry messages to be specific. By that, what we mean is basically messages that are generated using data on the edge are packaged up into an abstracted message format, and those messages are then sent to the IOT hub and the IOT hub can unpack those messages, and then for example take the payload and move them on to a pipeline of compute that you have defined in the cloud, right?

Also, we talked about multiple modules themselves. When you have multiple modules running on the edge, you want to be able to communicate between these modules. The edge runtime with this SDKs provides these paradigms called routes, which is very similar to if you're familiar with the network parlance with the networking routes. Really, what it's telling you is how these modules themselves are communicating and what are the inputs from one module, how does it feed into an – how does an output from one module feed into an input from another module and how do these paradigms really exist, right? All of this is part of the IoT edge runtime and allows you to weave this chain of compute very easily using the tools and the SDKs.

[0:41:12.8] JM: Where does Kubernetes come into this? If I want to have Kubernetes functionality in my edge deployment, where does that fit in? Where does it integrate with this other cloud specific platform software that we're talking about?

[0:41:29.4] VY: Right now, we just went into general availability last week. The very next thing on our plate is to really look at high availability. As you can probably tell from our conversation, people are starting to use these edge solutions for really mission-critical stuff as you talked about. There, a single point of failure is not an option. You cannot have one edge device, and if it goes down, your entire factory floor is down. That's a no-go right from the beginning.

There we are investigating Kubernetes for example, if you have a Kubernetes cluster, how can we leverage some of those capabilities to run your edge solution in an existing cluster? That's something that's coming up in our releases soon, right? What we currently have, Kubernetes support to be able to deploy, to be able to do cloud side solution and an edge side solution from a single pane of glass, so that is one example. Let me give you an example of that on how that works.

We talked about machine learning, right? Let's say we have some data that is on the edge and this is for example some camera data. This camera data is very secured and you want to keep it secure, so you have an encryption algorithm where you're encrypting this data before you're sending some key frames to the cloud where you're decrypting it. What you can now do is use Kubernetes to create, or to define a deployment that includes your cloud side components and your edge side components.

If you're encrypting data in the cloud – on the edge, excuse me, you also want to decrypt it in the cloud. Let's say you have just rolled a new version of the encrypted and then of the encryption algorithm that is a lot more secure. You can for example, use a project we've built using the virtual Kubelet, where you can define your edge site deployment and your service site deployment, or your cloud site deployment in one deployment manifest.

Using the Kubernetes console, or Kube CTL, or whatever tools you're using for managing Kubernetes in the cloud, submit this to a single cluster using a single pane of glass. Immediately, the portions that were on the cloud get updated with your V2 of decryption software, and the portions on the edge which are actually encrypting the software – sorry, encrypting the frames, get deployed to your edge devices, right?

You have both your edge side part of the solution and the cloud side part of the solution being managed as a single unit. That's really powerful, and that's what the virtual Kubelet allows us to do an integration with the virtual Kubelet allows us to do.

[0:44:11.3] JM: This is in contrast. This model of deployment that you just described, this would allow you to treat everything as one big Kubernetes cluster as opposed to having two separate Kubernetes clusters that are just communicating with each other?

[0:44:26.1] VY: Exactly. You got it. The way we do that is using the virtual Kubelet and further, and to give you a little bit of background on what the virtual Kubelet is, is pretty much a software abstraction of a Kubernetes node. Most Kubernetes nodes are backed by either bare metal, but in most cases a virtual machine, which has compute. The Kubernetes, the virtual Kubelet allows you to specify your Kubernetes node that looks just like a node to the Kubernetes cluster, but is backed by a third-party service. That third-party service in our case is the IoT hub.

When you have one deployment with – let's talk about the edge pieces themselves, you can then specify these particular deployment that I'm specifying, goes to the virtual Kubelet. Do not schedule this on ordinary Kubernetes nodes. You can specify that using various node selectors within the Kubernetes deployment itself. Then what happens is that particular deployment, or the part specification is sent to the virtual Kubelet, where it was meant to go. Our edge provider provides a translation service between the Kubernetes pod specification, to the edge deployment that I talked about before, and ships this edge deployment off to the backing IoT hub.

Now the IoT hub and already understands how to manage devices at scale, using for example the device selector and our for example, the candy factory where we talked about applying a deployment just to a subset of devices. Now, because it has now received deployment in its native format, it can then use that software specification and send it down to all the edge devices that are connected to it, right? That can be controlled by the device selector itself. That's how we achieve consistency between the edge deployments authored in Kubernetes, versus the edge deployments that are understood by the IoT edge runtime. Does that make sense?

[0:46:25.9] JM: Yes. Here, I want to recount what virtual Kubelet is. We've done a show where we talked in depth about virtual Kubelet. I know this lets Kubernetes connect to APIs other than Kubernetes. If you wanted to utilize serverless technology like it was a Kubernetes node and use the burst capacity of serverless nodes, for example, you could use the Kubernetes APIs while interfacing with serverless scalability using the virtual Kubelet. That's one example. You could also have the standalone container instances for longer lasting, longer lived compute nodes that you could burst out on to. Explain what the virtual Kubelet is in a little more detail and recount why that is important to this deployment.

[0:47:20.4] VY: Sure. Yeah, exactly. You got it right about the virtual Kubelet itself. Think of it as a node that is implementing the Kubelet API, Kubelet is just one of the – is an API server that is running within each node. Once this particular Kubelet is running on a node, the Kubernetes scheduler can schedule work to it. It can actually apply workloads to it, right? What the virtual Kubelet allows you to do is to back this virtual node by some other service, and these are all examples that you talked about. It could be the ACI, or the Azure container instances for serverless container instances, for example, it could be other serverless ones. I think, even Amazon has made one interface to go with their serverless container solution.

What we have done is have an implementation using the edge provider, where this virtual node can be backed by an IoT hub, right? In IoT land, interfacing this with IoT hub is useful because of the scale in one. For example, you could have a software configuration where you want to manage the configuration of hundreds of thousands of devices, creating Kubernetes nodes for each of these hundred thousands of devices doesn't really scale, right? Kubernetes at present is not designed to scale for this level of – this number of nodes, but the IoT hub is because that what was meant to do. That was one of the design principles right off the bat.

What we can do is bring both of these awesome worlds together using the virtual Kubelet, where you can do a whole bunch of operations using Kubernetes concepts. Then when you want to manage at scale hundreds of thousands of devices, you can offload that to the IoT hub, and you do that using the IoT hub deployments. Now you have the best of both worlds, where you can log into your Kubernetes cluster on in the cloud and start managing your edge devices, so you don't have to learn any new APIs, or any new technologies around IoT and IoT hub, or any of those technologies. You can use your existing knowledge on cloud deployment, but now start adding edge pieces to it as well.

[0:49:48.6] JM: Got it. In the cloud, you have you would have a Kubernetes deployment and the Kubernetes deployment in the cloud that you're operating against has a virtual Kubelet. The virtual Kubelet interfaces with the IoT hub in the candy factory, and the IoT hub is fulfilling the interface, the Kubernetes interface that you’re programming against in the cloud. You don't even necessarily need a Kubernetes deployment at the edge in that example, is that correct?

[0:50:24.0] VY: Absolutely. Yes, you don't. Yeah. As we know, Kubernetes deployments can be a little bit involved for good reason. You want high availability, you want to be able to spread, have these consensus-based systems being set up. Once you have an edge device, this is a really easy way to now not really need Kubernetes on the edge, but really get all the benefits of using Kubernetes to deploy software to it.

[0:50:50.0] JM: In that world, is there any advantage to actually having Kubernetes running at the edge?

[0:50:56.7] VY: Yes. One of the advantages is high availability, right? That's really the bread and butter of cluster schedulers, where if you are deploying onto a mission-critical edge infrastructure, you cannot have a single point of failure. What we have at the moment, that is something that we are building as we speak, but what is out there on most edge solutions out there are really focused on one edge device. You want to almost think of it as a service that you can deploy, just like you deploy to the cloud, you deploy to highly available cluster on the edge. Now on the edge, when you have those high availability scenarios, it may very well make sense to have Kubernetes be the scheduler that is managing that device, or that cluster.

[0:51:41.4] JM: Yeah. Yeah, that makes sense. I know we're near the end of our time. I want to get a little bit of perspective on the future. I showed my hand a little bit earlier about what I'm excited about in terms of the IoT stuff involving connected cars and drones and these sorts of things, and being able to control these things from software. Give me your perspective on the future. How far are we from that stuff that I described, or do you have a different picture of where we go from here and what IOT edge software deployments will look like in the near future?

[0:52:22.4] VY: Yeah, definitely. The way I look at it and the future is almost not tie it to just IoT at all, right? In the sense that if you think about the problem and at a larger scale, it's just that you have these huge amounts of data that is being generated on the edge and that's only getting more and more true as we progress into the future. You want to be able to apply compute to it. If you look at how CTRC puts it, it’s like that you have the intelligent cloud and the intelligent edge, that's it. It's not the intelligent IoT edge, it's the intelligent edge.

What I am excited about is some of the technologies and the platform, the edge runtime, the containerization, these various open ecosystem that is building around the intelligent edge to be almost a free-standing concept by itself. What we have currently is yes, in the context of IoT, but in the future as things become more capable, compute itself becomes more capable, being able to think of it in a broader terms to be able to apply compute to this vast number of systems that come online, right, on the close to where the data is. That is really exciting to me.

This is like the foray into it, but there are a whole bunch of other pieces like we talked about, where you have to talk about high availability, you have to talk about the other manageability of these things, security is a huge concern and how do we secure this. For that, we have things like Azure sphere and how we can integrate some of those concepts into the edge paradigm itself is something that's super exciting for me.

[0:54:01.5] JM: What are the kinds of tools around IoT? I mean, so we've outlined a couple different tools today; IoT hub, this thing that's a gateway between your cloud and your edge deployment, we've talked about the edge platform, we talked about virtual Kubelet, these are some very specific technologies. What are the kinds of tools doing – perhaps tooling around machine learning at the edge? What are the other areas of tooling do you expect to see in the next five years?

[0:54:33.8] VY: Oh, some of the tooling I would say first is to make it more accessible, the edge computing itself. Again, if you look at it from a developer perspective, this could be just extensions and tooling for building these modules, or this edge compute itself, and to make it really easy to be able to talk to the cloud. That's something we're investing a lot on. We have a whole bunch of Visual Studio code extensions with IoT edge that's only getting more and more important. We want to bring in other technologies like Visual Studio into the foray as well.

As well as more sophisticated debug tools, right? That debug and monitoring. These are tools that we feel can be improved, or need to be almost rethought in some cases to manage edge deployments. When something goes wrong, how do you first get notified that something's not functioning the way it should? What are the tools around that and what are the tools around debugging issues as they come up, right? These are two important things I think that we will invest a lot of time on and as an industry, we will need to look at as compute scales, not just in the cloud, but also these various devices on the edge.

[0:55:43.5] JM: Venkat Yalla, thank you for coming on Software Engineering Daily. It's been really fun talking to you.

[0:55:46.8] VY: Absolutely. Thanks so much, Jeff.

[END OF INTERVIEW]

[0:55:51.3] JM: Test Collab is a modern test management solution which integrates with your current issue manager out of the box. For all development teams, it's necessary to keep software quality in check, but testing is complex. There are various features to tests, there's various configurations and browsers and platforms. How do you solve that problem? That's where Test Collab comes in. Test Collab enables collaboration between software testers and developers. It offers wonderful features like one-click bug reporting while the tester is running the tests, collaboration on test cases, test executions, test automation integration and time tracking.

It also lets you do a detailed test planning, so that you can configure platforms and browsers and configurations or any variables in your application and divide these tasks effortlessly among your team for quality assurance. All of the manual tests run by your team are recorded and saved for analysis automatically, so you'll see how many test cases passed and failed with plenty of QA metrics and intelligent reports, which will help your applications' quality.

It's very flexible to use and it fits your development cycle. Check it out on testcollab.com/sedaily. That's T-E-S-T-C-O-L-L-A-B.com/sedaily. Testcollab.com/sedaily.

[END]