EPISODE 470

[INTRODUCTION]

[0:00:00.5] JM: Cloud computing changed the economics of running a software company. A cloud is a network of data centers that offers compute resources to developers. In the 1990's, software companies purchased servers. This was an upfront capital expense the required tens of thousands of dollars. In the early 2000's, cloud computing started and it turned the capital expense into an operational expense. Instead of a huge bulk purchase of servers, a software developer can now pay as they go, and this is so much easier on the budget than the bulk purchase.

This transformation of capital expense into operational expense significantly lowered the barrier to entry to starting software companies. Cloud computing created an economic boom that continues to today and shows no signs of slowing down. Of course, the upfront capital expense did not disappear altogether. It got aggregated into the cloud providers. Instead of individual software companies buying servers like they did in the 90's, cloud providers buy thousands of servers and build data centers around the world. Because of the aggregation, cloud providers get economies of scale. It's much cheaper for a single cloud provider to buy 1,000 servers than for 1,000 software companies to buy one server.

The first wave of cloud computing was all about deploying our applications to servers that run in cloud data centers. The next wave of cloud computing is moving more and more of our application logic into managed services.

In the first wave of cloud computing we had to spin up a server to run our database. We needed a server for search engine and our load balancer. The logical unit of a server comes with the risk that that server will crash or will be hacked or will develop a bug that is difficult to solve and managed services abstract away the notion of a server. Database as a service, search as a service, load-balancing as a service; these services are reliable Lego blocks that we can build entire applications on top of.

Developers pay a premium for a managed service, but they're happy to pay that premium because it represents a promise that this service will not crash. It will not be hacked due to a software vulnerability that has been patched by everybody but you and it's not going to develop bugs that are your responsibility, because it's a managed service.

Developers are very happy to pay higher unit prices for these managed services than they pay for raw servers and cloud providers are very happy to develop and offer these managed services because it turns the infrastructure as a service business into the software as a service business. Software as a service has better margins, better retention and better competitive differentiation than this infrastructure as a service business. Of course, the cloud providers actually are offering infrastructure and software as a service, but the managed services are often an additive higher-margin business that they can build.

As a developer, I'm actually looking forward to building applications completely out of specialized managed services, but we're still pretty far from that time. Developers today still need to write their own custom application logic and their own backend services, but there are a set of tools that allow developers to write these services, their own services while getting some of the resiliency and the scalability of the managed services. These tools for getting the resiliency of a managed service while getting to write your own code are functions as a service and Kubernetes.

Functions as a service let developers deploy stateless application logic that is cheap and scalable. Functions as a service still have some problems to overcome in the areas of state management and function composition and usability and developer education, but the future of functions as a service is quite bright.

Kubernetes on the other hand is a tool for managing containerized infrastructure that is fully ready for adoption by lots of people. Developers put their apps into containers on Kuberntes and Kubernetes provides a control plane for deployment and scalability and load-balancing and monitoring. It's basically all of the things that you would want out of a managed service, but it's your own code. So things have become a lot easier when they're deployed into Kubernetes, and this is why Kubernetes has become so popular and it's also why Kubernetes itself is being offered as a managed service by many cloud providers including IBM. For the last decade, IBM has been building out its cloud offerings and for two of those years Jason McGee has been CTO of IBM cloud platform.

In this episode, Jason discusses what it is like to build and manage a cloud from operations, to economics, to engineering. It was a real treat to have the CTO of an entire cloud platform on the show, and we've done some other shows about engineering cloud services at Microsoft and Google and Amazon and Digital Ocean, basically all of the cloud providers. You can find these old episodes by downloading the Software Engineering Daily app for iOS and for android. You can of course find many of our episodes in podcast players, but in those podcast players you can actually only access the most recent 100 episodes and with the apps, you can access all of our old episodes, you can search over our content, you can find content that is related to each other, and we're building a new way to consume content about software engineering. These apps are also open-sourced at github.com/softwareengineeringdaily. So if you're looking for an open source project to get involved with, we would love to get your help.

With all that said, let's get on to this episode with Jason McGee.

[SPONSOR MESSAGE]

[0:06:30.8] JM: For years, when I started building a new app, I would use MongoDB. Now, I use MongoDB Atlas. MongoDB Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running.

MongoDB Atlas is the only database as a service from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous back-up, VPC peering, monitoring, and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy-to-use service. You could forget about needing to patch your Mongo instances and keep it up-to-date, because Atlas automatically updates its version.

Check you mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free. Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature.

Get started with a free three-node replica set, no credit card is required. As an inclusive offer for Software Engineering Daily listeners, use code sedaily for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out.

Thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW CONTINUED]

[0:08:29.6] JM: Jason McGee is the VP and CTO of the IBM cloud platform. Jason, welcome to Software Engineering Daily.

[0:08:35.9] JMG: Thanks, Jeff.

[0:08:36.7] JM: I want to start by talking about cloud in a broad sense. We've had the cloud for more than a decade at this point that just the word cloud computing. We've had things like cloud computing before that, but it really started getting formalized and start getting a lot of traction I think about a decade ago. How have cloud workloads changed over that period of time in the last 10 or 12 years?

[0:09:03.1] JMG: Yeah, I think they've changed pretty substantially. I think in the early years of cloud — On the early years of cloud, I think people mostly were trying to get their head around what the model is all about and used it for some very specific scenarios, like burst capacity or public facing applications within their companies.

I think as cloud started to get adopted in earnest, people viewed initially as a cost-savings platform. I was all about efficiency and cost savings and being able to run their core infrastructure at a lower price point.

What I think has happened in the last few years is that mindset has completely flipped and most enterprises today are looking at cloud computing as a vehicle to help them move faster, to help them innovate and help them deliver new capabilities, to help them compete in the markets that they're in by having more agile access to capability and by allowing them a platform where they can try things out and experiment at a relatively low starting cost.

[0:10:04.8] JM: Many of the shoes that I've done have been with newer companies that started on the cloud, and for these companies I think the contemporary term might be cloud native, but there are lots of older companies that were around before the cloud was a thing and some of the ones I've talked to are hybrid cloud. So they have their own on-prem infrastructure. They haven't moved off of it, and they use the cloud as an augmentative tool.

These hybrid cloud companies, what do they do for the cloud and what do they do with their on-prem infrastructure. How does that division of labor between their resources manifest in the hybrid cloud model?

[0:10:48.5] JMG: Yeah. I think those older companies which many of them that kind of established success companies, the history of IT in general for them has been one of evolution and new technologies being blended with existing infrastructure and investments that they've made. To some extent, hybrid for them is no different than what they've always done in IT which is look at new developments, look at new technologies and approaches and figure out how best to adopt them and use them for a leverage within their companies.

I kind of see kind of two predominant patterns emerging within those hybrid enterprises. One is they use cloud as a platform for building new things, to adopt the latest technical approaches in architectures and use it as an innovation platform for building new solutions. In those scenarios they of course use things like public cloud and it's really important for them to be able to connect back to the investments that they had. They're hybrid in the sense of connecting cloud-hosted applications with existing systems.

The other scenario which I see pretty commonly with them is what I kind of call modernize and extend. They look at cloud both on-prem and public cloud as a way to re-host existing applications without rewriting them to get some efficiencies, some improved velocity around dev-ops for those existing apps and then extend those apps with new experiences.

A simple example is like the revolution that's happening now around AI and cognitive, and there're lots of companies who want to add cognitive capabilities to those existing apps, but they're not going to start over and write them from scratch, and so they use hybrid as a kind of modernization platform.

[0:12:33.0] JM: I've done a bunch of shows with companies about how infrastructure has change from the point of engineers who are utilizing the cloud, and for these engineers, work has gotten a lot easier over the past decade largely due to managed services even over the last two to five years. I just had a conversation with somebody from Thumbtack, which is a marketplace company that's really taken off from the last couple of years and talking to them about how their rapid growth, their hockey stick growth in the last couple of years has manifested and the level of operational difficulties that they've had. It's much easier than the operational difficulties of a company like Uber that hockey sticked maybe five years ago. Even in the last three or four years, the managed infrastructure, the tooling perhaps of containers I'm sure we'll talk about that, and just — Even containers and the ways that many people use them require a little bit more operational burden than something like a managed database service or functions as a service.

This is all talking about cloud infrastructure from the point of view of an engineer, and you're the CTO of a cloud itself. I have a unique opportunity to ask you some questions about managing a cloud platform. The first thing I'm curious about is; does that translate to managing a cloud platform itself? Because obviously if I'm an engineer building an application level thing, like a marketplace, things have gotten a lot easier. But if I'm writing the code that is managing resources across a datacenter and is the underlying guts of a managed service, has my life gotten easier as well?

[0:14:23.6] JMG: Yeah. Yeah, I think it has actually. I think the macro trend that's been going on for a number of years that cloud has certainly accelerated is the shift from when you're going to solve a problem, having to think about the infrastructure to support that application. That was kind of the original model, like I want to solve some problem. The first third of the project is like, "How do I build up all the infrastructure I need to get started?" That has been flipping to the inverse, which is, "I can just focus on my application and all the resources underneath my app are kind of at my fingertips, either as managed services or as API provisioned infrastructure resources," and that has enabled people to start with lower risk and to be much more agile about how they solve problems. I think that transition applies to me as well as a cloud service provider.

It depends on what layer of the stack we're talking about. Obviously, the closer we get to the bottom, maybe the less that managed approach, that API-driven approach helps, because you're the guy providing the APIs and providing the management But if you think about a comprehensive platform like IBM cloud, at different layers of the stack, absolutely, it helps us.

I have teams like the Watson building who's building AI capability. They are riding on top of our container platform in cloud. For those engineers, building those cloud services, they don't have to think about infrastructure and they don't have to think about how to do availability and scale in the same way. They can simply ask for a managed container environment and focus on building cognitive APIs that we expose as part of our cloud. I think that same trend that's helping kind of end-user application developers is also helping me and my team built the cloud.

[0:16:12.0] JM: You got these datacenters all around the world that are managing the software that provides these managed services and provides the ability for developers to provision VMs or provisions containers. Give me an overview of the software that is used to manage and provision these clusters themselves.

[0:16:35.1] JMG: Yeah, sure. The exact architecture in software that's used to manage some portion of the cloud is a pretty specific to the problem domain, right? Whether we're doing VM management or storage or containers or databases, a lot of that is frankly a combination of open-source technology, which as you know IBM is a big believer in contributor to and custom software that we've written, because frankly the problem that I have as a cloud provider is very different than kind of the typical application developer problem. I have to provide services that are managed at scale with isolation and multi-tendency. That isn't really an off-the-shelf software problem, and so we've built stacks that are appropriate for the different services that we host.

In many cases, we've reused the things that we expose for developers. So for example, my team runs our Kubernetes-based container service on IBM cloud and we actually use Kubernetes to manage Kubernetes, because the core problem of how do I provide a scale highly available platform to host application logic is the — I have that problem in my control plane, just like a customer might have that problem in their application space.

We tend to reuse a lot of the same services that we're building for customers to manage our own stuff. Now, we have to do that globally and we have to operate that, and so we also have adopted SRE operations principles. We've build a lot of automation. We've adopted things like ChatOps and AI technologies as part of our operation stack to help us do this at scale.

[SPONSOR MESSAGE]

[0:18:23.5] JM: Dice helps you accelerate your tech career. Whether you're actively looking for a job or you need insights to grow in your current role, Dice has the resources that you need. Dice's mobile app is the fastest and easiest way to get ahead. Search thousands of tech jobs, from software engineering, to UI, to UX, to product management.

Discover your worth with Dice's salary predictor based on your unique skillset. Uncover new opportunities with Dice's new career-pathing tool, which can you give insights about the best types of roles to transition to and the skills that you'll need to get there.

Manage your tech career and download the Dice Careers App on android or iOS today. You can check out dice.com/sedaily and support Software Engineering Daily. That way you can find out more about Dice and their products and their services by going to dice.com/sedaily. Thanks to Dice for being a continued sponsor, and let's get back to this episode.

[INTERVIEW]

[0:19:39.2] JM: Is there anything like open stack of Mesos that does the job of managing the clusters at a lower level than Kubernetes, or is Kubernetes kind of the good enough a system for managing a global network of datacenters? I guess I'm wondering if there's some piece of software that has the God's eye view of all the different datacenters, because my understanding was like Kubernetes, you might use Kubernetes to manage perhaps one data center. I'm not sure about a network of datacenters, or maybe I'm just thinking about the abstractions wrong.

[0:20:16.9] JMG: Yeah. There's no one single God's eye view of the whole planet. Below containers at the infrastructure layer, we obviously have an infrastructure control plane that's managing the datacenters, that's managing bare metal servers and virtual machines and networks within those data centers. That is a largely custom control plane that's custom built for the problem at hand of running a global scale public cloud platform.

Kubernetes though is actually one of the fundamental building blocks that we use and is the base layer that we build most of the IBM cloud on. Not only is it a service that we make available to developers who use our platform, but it's actually the underpinnings of our identity and access management systems, it's the underpinning of our cognitive data services. It is the kind of common layer that everyone builds on and has many of the characteristics to kind of run the cluster if you will underneath a whole collection of workloads.

[0:21:22.7] JM: When Kubernetes came out, was there some significant refactoring to be done to replace the underlying infrastructure that was before Kubernetes or do you just leave that stuff running in its previous manifestation and you spin up a new stuff on top of Kubernetes.

[0:21:42.5] JMG: Yes. One of the great both challenges and advantages of cloud is this kind of pace of innovation that underlies it. We update our environments continuously, and when new technologies emerge or we help them emerge, like Kubernetes, we absolutely go and change our architectures and change our implementation as appropriate to take advantage of and to expose those capabilities to user users.

None of those underpinning are static if you will or left around. Now, the mechanics you're kind of leaning a little bit on, well how do you actually do it? Absolutely, we use A/B testing and Canary deployment models to be able to kind of roll in new capabilities sand new implementations of existing capabilities in a non-destructive way. It's one of the challenges of being a cloud provider is you have to do everything while the bus is moving, while everything is up without disruption. As we would bring in something significant, like Kubernetes, yeah, we'll do that side-by-side for a period of time as we transition over to some new architecture that we're going to run.

[0:22:49.2] JM: You mentioned the adoption of SRE style operations. Talk about the operations model of the IBM cloud, like what's the division of labor look like. Are these SRE people sitting in the data center itself or are they teams that are remote from the datacenter? Yeah, I guess talk about just the operations framework and the management structure.

[0:23:18.2] JMG: Sure. Generally speaking, the way we describe it is you build it, you run it. Meaning if you think about a cloud platform like IBM cloud, it's comprised of a collection of services and each of the services is owned by a team and that team owns the lifecycle of that service end-to-end. Meaning they build it and they run it. They operate it.

Our SREs are, generally speaking, aligned with each of the services and they sit conceptually with the service, not in the datacenter, but in the development team. SRE is a model where you're applying software engineering practices and principles to the domain of operations, and so logically those SREs are aligned with the development organizations and can sit in this part of those teams. Their jobs is to run that service and to design it for availability and to continuously improve how we deploy and operate and update and recover that service though automation and through software technology.

We of course have some centralized functionality around SOC and NOC and central learning and other things to help kind of wire the cloud together. On the whole, we kind of run each service as its own end-to-end lifecycle, and that's really powerful and it's actually a model that a lot of customers are trying to adopt. It's a model that I think is aligned with this transition that I've described that we're becoming more application oriented. It's makes sense that you run things from the perspective of that application or service, whatever word you want to use. Therefore, that operations functions tends to align itself with the capability itself.

[0:24:51.8] JM: How rapidly do you have to grow the size of an operations team relative to the size of your customer base? Because I can imagine being able to scale your employee operations-base at a kind of a linear rate, and being able to scale customers at an exponential rate, I think it seems like one of the great economies of scale of being a cloud operator. Is my intuition correct there or am I off?

[0:25:22.5] JMG: No, I think your intuition is correct. Obviously, the great trick of running these kinds of systems is making your operations growth and your customer growth not be parallel with each other, or in the worst case, your operations cost grows faster than your customer-base, which is death to a cloud service, and so we spent a lot of time.

Part of the reason that we use an SRE methodology is to ensure that we're always looking towards how do we run these services, how do we deal with their growth as efficiently as possible so that as consumption scales, the cost of operation around that service doesn't go up linearly with it. How that's accomplished varies a lot by service, but fundamentally it's rooted in deep automation, in deep instrumentation of the environment. We actually have a sense for what's going on at all levels, not only operational status, but performance and user behavior and using that data to drive the next round of development changes and how we automate the operations to that platform, and that's something that we work on every day.

[0:26:24.3] JM: On the developer side, containers have become the deployment unit of choice. How does the growth curve of containerized workloads look over the last five years? Do you have any numbers or any insights on just how much of the industry has adopted containers and what they're using those containers for?

[0:26:47.0] JMG: Yeah. I don't know that I have great numbers off the top of my head. Certainly, customer interest in containers has been off the charts. I think the highest of any new technology in certainly my 20 year career in IT, it is used essentially every customer I talk to, they have containers and are using containers. Now, where they are in their journey has varied. I think one of the big transitions we've been seeing in the last 12 months maybe, maybe even less, maybe nine months, is really the transition of people using containers early in the lifecycle for development. So they're using it on their development teams. They're using it for packaging and testing, but then they were deploying using kind of traditional models. Maybe they take those containers and they put them in virtual machines and they deploy them with a traditional virtual machine model.

In the last nine months or so, they've started to make the transition to actually using containers as their production operations methodology. Meaning, they're adopting things like Kubernetes that the unit of deployment is a container or a collection of containers, not a VM, and it means that they're starting to take advantage of the network architectures and operational monitoring that containers bring, and the pace of that has been just really tremendous. Like every year that goes by, it's probably double where I thought it would be that previous year.

[0:28:08.9] JM: How does the growth of adoption containers change the economics of being a cloud provider?

[0:28:15.1] JMG: It's interesting. I think containers are one of those technologies that part of the reason that I think their growth has been so tremendously high is that there's a value prop for containers for kind of all constituencies. It helps developers build software and delivery software faster. It helps operations teams on the kind of customer side be more efficient and achieve higher availability. As a cloud service provider, it helps economics for us as well. Because, fundamentally, a container-base model allows the platform to be more dynamic so you can create and delete things more quickly and allows the environment to be consumed at a higher density.

At its heart, part of the cloud economics model is density and sharing of resources and containers allow the resources we've deployed in the cloud to be utilized at a higher rate and has enabled new models. Like you mentioned earlier, things like serverless or functions. At their heart, functions or serverless platforms, are essentially container platforms. If you look at how they're actually built, internally they're container platforms. They have a different execution model and API, but they're fundamentally container platforms. Because of the advantages of containers, you actually can get to an economic model, which is literally paying for CPU cycles with no idle time, and that isn't really possible with other technical approaches. I think those technologies have really enabled new economic models to emerge in the cloud.

[0:29:47.7] JM: This is basically because if I provision a virtual machine, it's going to take up this large block of space and I'm probably not going to utilize that space to capacity. Whereas a container is typically a slice of a virtual machine. You're going to take a virtual machine, slice it up into a bunch of containers and those containers are provisioned with a more accurate fraction of the overall resource and it more closely aligns with what you actually need so that more efficient provisioning is a more efficient usage of resources.

Did IBM have expertise, like internal expertise in containers prior to the popularity of Docker and the mass adoption of containers? Were you using them internally?

[0:30:42.3] JMG: Yeah, absolutely. In fact, one of the interesting things about the history of containers is that containers are not new. Containers are 12+ years old and their origin goes back to work in Linux and in Unix. IBM was one of the lead contributors with early work that we did in Linux and in our AIX, our power platforms. Some of the ideas like name spaces that are popular in the container space today are ideas that IBM helped create. Our history goes back to actually helping be one of the people in the industry along with Google and Sun and others who actually created all the underpinnings that Docker then put together.

Docker's hat trick was taking all of these powerful technology that are built into the Linux kernel and making it accessible for the average developer.

[0:31:32.3] JM: And added a logo.

[0:31:34.0] JMG: Yeah. Well, making it consumable. They made it so that someone who is not a Linux kernel expert could figure out how to leverage this technology to advantage for them. The core capability is stuff that we help build and stuff that we had used in various capabilities that we've built overtime, both the work we've been doing before Docker and Kubernetes in the cloud foundry space or back further into our main frame in Unix platforms. We have a long history of kind of taking advantage of these capabilities and how we ran things, and what's happened I think in the last few years is just the democratization of this technology in a way that means that it's now the defecto way you build everything. There's no project I think that starts today that doesn't just assume they're going to use something like containers as the platform to build that app.

[0:32:24.9] JM: The big thing about Kubernetes was it gave people this single tool to manage their different containers. It lets you, for example, look a collection of containers together as the abstraction of a service and you could just start to create rules around how that service scales up and down, the underlying containers are scaling up and down. Of course, to deploy and manage Kubernetes for some people has been a burden, and they don't want to do it, so they decide to use a container platform and all of the cloud providers are building their container platforms to give a fully managed version of the container platform, and these things are typically build on Kubernetes.

Can you help explain what are these operational burdens that people have from self-managing Kubernetes that tend to get remedied when they move to a container platform that's fully managed by the cloud provider?

[0:33:28.0] JMG: Yeah, great. Yeah, it's a great question. I think it goes back to your comments in the beginning about the value of kind of managed services in cloud. I think the problems that a managed container service solve for the user, they range. There's the obvious. The mechanics of installing, connecting and configuring a collection of resources into a functioning container cluster, that's not easy or free. That takes work. It takes knowledge, and managed services like that and cloud container service do that for you. They literally give you single API call, "Wait a minute or two. You have a cluster up and running."

That's a day one experience, right? Once you have cluster — If you always keep in mind at the end of the day, what people want to do is build and run applications on that cluster. Anything you can take away from them that they don't have to worry about anymore is an advantage. If you think about beyond day one, you have things like how do I add or remove capacity from that environment. How I recover when failures happen? Maybe the thing that people really don't recognize is keeping up with the pace of change. If you look at containers, Docker, you look at Kubernetes, you look at the surrounding technologies, they're moving at just a blistering pace. The Kubernetes has releases every few months. All the underlying technologies are constantly updating. Keeping up with that pace in the industry is really hard if you do it yourself. You need like fulltime people who are connected to the industry and in the communities to kind of know what's going on to keep up with those changes to make the associated configuration changes.

One of the advantages of a manage service is all of that is just done for you, and you can just focus on your apps. Me, personally, my career has kind of always centered around what we use to think of as the middlewar-based. I was a Java Webster guy for a long time.

The mission statement is kind of the same. It's like we're going to provide, in the cloud now, all of the building blocks you need to build your apps so that you don't have to worry about all the plumbing. A managed container service at the end of the day is plumbing to help you host your applications.

The thing that's cool about them is they solve some really hard problems, like availability and scale and networking in a way that's common across different programming languages and different application patterns. To me, that's the advantage, is like I can just focus on my apps.

[SPONSOR MESSAGE]

[0:35:57.7] JM: Do you have a product that is sold to software engineers? Are you looking to hire software engineers? Become a sponsor of Software Engineering Daily and support the show while getting your company into the ears of 24,000 developers around the world. Developers listen to Software Engineering Daily to find out about the latest strategies and tools for building software. Send me an email to find out more, jeff@softwareengineeringdaily.com.

The sponsors of Software Engineering Daily make this show possible, and I have enjoyed advertising for some of the brands that I personally love using in my software projects. If you're curious about becoming a sponsor, send me an email, or email your marketing director and tell them that they should send me an email, jeff@softwareengineeringdaily.com.

Thanks as always for listening and supporting the show, and let's get on with the show.

[INTERVIEW CONTINUED]

[0:36:59.8] JM: The managed container platform that you built at IBM is called IBM Container Service. Can you describe the design process for this project?

[0:37:10.0] JMG: Yeah, sure. The IBM Container Service, as we've been alluding to, is a Kubernetes-based service. The design process was really honed in around that basic storyline that I just gave you. How can we provide the easiest user experience around creating and managing a container environment that our customer can use to host their apps?

Then what hard problems can we solve for them beyond just giving them the environment? An example could be security. Maybe the unspoken reality is that most developers knows that security is important and most of them don't want to think about it. If we can do things in the platform to make the environment secure out of the box to give people visibility into the security posture and vulnerability of software that they're developing, that will help them be more successful at deploying secure software without having to become security experts.

On the basis of this kind of core Kubernetes environment we can stand up, we thought about in that design process what are the additional problems that are hard that we can solve? Security. Another one is if you look at cloud-native apps, one of the design principles of cloud-native apps is you deploy your application across a multiple availability zones in multiple regions around the world and that's how you achieve higher availability in the face of a failure of one zone or one region in the cloud.

On paper, that's easy to say, but in practice that's hard. It's multiple deployment pipelines. It's global traffic managers and DNS checks to distribute traffic. We said, "Can we automate that? Can we have the service help you automatically set up highly available globally distributed clusters so that as an app developer you can just deploy your app into these environments and those problems are solved for you.

That was our design process, was give them the core capability they need to host their apps with Kubernetes, and then around that environment solve the hard problems that all developers have to solve to build production-worthy scalable cloud-native software.

[0:39:12.2] JM: Okay. You mentioned security. You had a project recently. You partnered with Google on called Grafeas, which does — This is like audit and compliance of a containerized software platform. Explain what that means.

[0:39:28.7] JMG: Yeah. One of the things that kinds of emergent in the container space is this idea of kind of a secure software delivery pipeline or supply chain. I think it's a manifestation of the fact that because containers have kind of normalized things like packaging, and because Kubernetes has normalized how you launch applications and make them available, you can actually start to define the steps that you go through to get a change, checked into a source code control system and delivered all the way your production.

Along that pipeline, there's a lot of metadata that's generated and you want to use that metadata to make decisions. A simple example; in IBM cloud, as part of our container service, we have a tool called learnability adviser. What that does is scan all of your container images and it looks for things like vulnerable software packages that have known CBEs against them and it looks for poor configuration. You have weak password rules or you don't have password expiration or you have SSH enabled, and it generates a report of the vulnerability status of that software. If you think about a delivery pipeline, you might want to be able to say things like, "In the production environment, no applications should get deployed that has a vulnerable image or whose image has not passed the vulnerability scan."

Now, we can do that in a point-wise way, like we integrate all of these tools. If you look at the container space, there's just a tremendous wealth of point solutions that look at different parts of these problems and they don't know how to talk to each other.

The Grafeas project that we've been doing with Google and others is really about solving that. It's like can we all agree on a common way to share all of these metadata that's being generated in the delivery pipeline of software and use it to make policy decisions about whether we're going to deploy something or not. Grafeas actually has two pieces. It has this metadata basis, essentially, where we can all share the information, and then it has an enforcement component that runs inside Kubernetes, for example, that looks at that data and applies policy to it to decide whether this is something we want to allow.

To me, this is the exciting part of containers, is like some people think of containers as kind of more efficient virtualization, but in reality it's completely changing how development life cycle happens, how the development process works, how we build applications. I think that's pretty exciting.

[0:41:54.1] JM: Yeah, the other project that comes to mind that you worked with Google on is the Istio project, and this is another fairly new project. We've done a show about this. We're doing another one soon. Istio, as I understand, is it's a service mesh and it's built on top of what is called a service proxy. Envoy is a service proxy that was originally built by Lyft, because Lyft had this problem where they were afraid to deploy their software on any individual container, because they were afraid of all of the interrelations between different services that were dependent on an upstream of a downstream container. Basically, what they said is like, "Okay. We've got all these containers that are running. What if we try to standardize some things around how these containers interact with each other so that there's less sensitivity when we deploy a container?"

Then Envoy had such success that this Istio project came up around it and Istio is sort of like, "Okay. Now that we've got some standardization among these different services that have a proxying layer, we can now create this mesh," that I think if I recall correctly is kind of gives you a control plane. It gives you a way of leveraging the fact that now you've got this side car running alongside all of your services that are doing proxying and doing all kinds of other things and that the Istio service mesh gives you a centralized area where you can do things that leverage having a side car and all of those different service instances. Am I articulating that correctly?

[0:43:39.6] JMG: Yeah, you are, and maybe I can give a little context. First off, I think Istio is one of the most exciting projects in the industry today. In fact, I was looking at some early kind of adoption data on the GitHub project and I think it's actually trending ahead of where Kubernetes was at the same point in its lifecycle.

Istio, I think — I think you got some of the history pretty spot on there. Actually, it's interesting, both my team at IBM, I've been doing work around this idea of the service mesh and kind of programmable routing control plane between different microservices. Google had been doing some work in parallel. They had been really focused on like security and telemetry, like visibility into what's happening between microservices. Then Lyft had this really awesome proxy component that had all the smarts of how to handle the traffic.

The real problem that Istio is trying to solve is as you adopt microservices and you go from having one thing to having hundreds or thousands of components, you need some way to kind of both understand and control what's happening between them. What we had seen happening for early adopters on microservice architectures was they were building some of the things that Istio does into each microservice. They would build monitoring agency and they would put configuration into control who they route it to. They would solve problems in a kind of custom way per service.

That of course puts a burden on each microservice team. It means every service solves the problems at a different way. It makes it really hard to change tools or get global visibility. What we decided was there's a need and a value in having a smart inner connect layer between the services that abstracted all of those kind of common communication concerns out of the microservice and in to the smart mesh and to do that in a way that was totally like programming model and language neutral. You can use it in whatever programming language you want. You don't have to write to an Istio API when you build your apps, you just use normal TCP and HTTP communication and we can kind of insert ourselves.

With that, I think Istio really gives people a lot of power in controlling how their services deploy and interact, even simple things. Like one of the most common things a development team has to do when they're doing microservices is delivery an update to production. Even in Kubernetes, by default, when you delivery a new version of an application, a pod into the cluster, as soon as it's running, it starts receiving traffic and you don't have any control over that. It just starts receiving the traffic.

Maybe what you want to do is decouple when you did the deploy from when people actually started using it. You could deploy it, validate that everything looks good and then maybe gradually roll people on to it. That's actually a pretty common scenario and that's kind of thing that Istio makes super easy, because which version of a service somebody routes to is not a function of the deployment operation. It's a function of a policy statement in the Istio control plane. You can deploy version three of a service. Everyone still uses version two, and then you can go into Istio and say, "Hey, send 50% of the traffic to version three, or send all of Jeff's traffic to version three and let him make sure that it looks good before I start to roll that out to others." That kind of architecture, that service mesh architecture I think is solving some really practical problems that people have been grappling with as they thought about, “How do I adopt microservices?”

[0:47:19.2] JM: For some people, this might be challenging to understand, because they might be thinking. “Okay. I thought Kubernetes was the thing that like manages all these containers, and now you’re giving me this other thing that manages all my containers. Why aren’t we just bundling all —” Okay. We’ve got — I’ve got these notes in front of me. I was at Qcon a couple of days ago and I saw a presentation from Louis Ryan who started the Istio project and he’s like, “Okay. Yeah, Istio gives us — Let’s see, observability, resilience, traffic control, security policies.” He’s like, “Well, I thought Kubernetes gave me this God’s eye view of my containers?” Why do you need another project? Why is this a good abstraction to throw into the mix?

[0:48:04.0] JMG: Yeah, it’s a great question, and I think if you look at it in detail, they’re actually very complementary. One of the interesting decisions that Louis and my team made together was while Istio stated objective is to work across different environments, not just Kubernetes, but Docker and Mesos and VMs and bare metal servers, because the reality is in a microservice architecture, you might have all those things.

What we decided on day one was to deeply integrate it with Kubernetes, and if you look at the problems that are being solved, they actually don’t overlap. They’re actually complimentary. A lot of the core underpinnings of Kubernetess are used by Istio. Kubernetes handles the basics of service discovery and handles the basics of routing of traffic between containers inside the cluster. Istio doesn’t really override that. What it does is its builds on top of that and adds in things like Kub doesn’t do. Like Kubernetes will handle if my container fails, restart it. Kubernetes will not handle what happens when service B fails and service A depends on it. What happens in service A when its dependency starts to misbehave? You might want to fail back to an alternate source or you might only want to wait 10 seconds for that failure to occur before going on. That kind of cascading failure problem, which sometimes people describe as circuit breaking. Those concepts are not built into Kubernetes. Version routing, it’s not built in to Kubernetes.

You could argue, we could just add all these to Kubernetes, and that’s kind of what Istio did. If you actually look at the architecture of Istio, it actually sits inside of Kubernetes as API extensions to Kubernetes. It doesn't actually feel like you installed an alternate orchestrator, it just feels like you extended Kubernetes with some new capabilities to solve this specific domain of interaction between microservices. I think they're actually quite complementary.

[0:50:00.1] JM: I want to jump forward in talking about serverless a little bit, the functions as a service. I interviewed one of your colleagues, Roderickk Rabbah who helped architect the OpenWhisk project. We had a pretty detailed discussion about serverless that will air before this show.

One thing I'm curious about, since you're in this CTO position, you are kind of like looking several steps ahead of where we stand today as people that are using clouds and deploying and managing clouds themselves. Do you think functions as a service, is this an abstraction that's going to be around for a long time, or do you feel like the functionality, the function as a service idea — I can imagine a world where this just gets absorbed into other managed services, like if you need to have your object storage respond to an event, it feels kind of weird to spin up a function as a service to respond to that event rather than having that code tightly coupled with the object storage itself. I'm not sure if that question makes sense, but I guess the bigger question is do you think function as a service is actually going to be something that's around for a long time?

[0:51:14.2] JMG: I think the idea of an event-driven function execution absolutely will be around for a long time. In fact, it's been around for a long time. The base paradigm of the kind of serverless movement has existed for a long time. Like it's basically how all UIs work. They're all event-driven function handlers.

I think what will change overtime is like my view has always been that the serverless platforms, the reason they grew up — Or I shouldn't say grew up, but are growing up, in the context of cloud, is because the value of those platforms I largely derived from the integration and events that flow between all of the other services in cloud. Used object store is an example. That's a really common use case for event systems of functions as a service, which is like, "Something changed in my object store bucket. Do something about it."

The user experience is you're right, are kind of divorced at the moment. You go to a function service to define a function that responds to an object storage event. You could just as easily go to object storage and say, "When this event happens, do X."

That's just a user experience thing. Like under the covers, that's kind of the same thing. I agree with you. I think it will tend to get embedded in the platforms, like in the overall cloud platform.

[0:52:31.4] JM: You can imagine that being more efficient to provision and manage if you've — I don't know. Actually, I know nothing about that, so I could be wrong about that.

[0:52:41.2] JMG: Yeah, I don't think it's provisioning efficiency. I think it's like how do people think about the world. I think in the use case where it's like, "When X happens here, I want to do something." It logically makes sense for you to be able to define the function there in that context.

There's other cases where maybe you're doing something more complex or you're doing a composition of a sequence of events, in which case you don't think about any one source. You kind of think about the sequence of events, like going to a functions platform and defining an application which is a sequence of functions that respond to a chain of events makes sense as a developer.

I think you need both, under the covers that really driving the same infrastructure, but you want to expose them. We're in the early stages of this movement. So in the early stages, I think it's more natural to just build a discreet thing. You build a function service and overtime it will get more embedded.

I also think you see that same split with functions and containers. Like today, functions, like OpenWhisk and containers with Kubernetes feel like two parallel things that sometimes are in opposition with each other. If you talk to true believers on both platforms, they'll tell you that all problems can be solved with containers or all problems can be solved with functions, right? I think the reality is all problems will be solved with a combination of functions and containers, because they're good at different things.

One of the things I've been really focused on in my role is how do we think these things evolve and how do we bring them together so they feel like a single cloud-native platform where I can use an event-oriented functions paradigm for some things and I can use containers for others. As an example, we have a client who's a bank and they have an online banking application. The core of that online banking system is a web application that's naturally fits into containers and they're running in containers, but they have the certain function of that app which is to do digital check deposits. If you think about a digital check deposit, it's a very event-oriented activity, like a new check arrives, like image of new check arrives in object storage, do image correction, do OCR, make a deposit to the backend system, send a confirmation email, go away.

It's very spikey in nature, like you get a lot of check deposits on Fridays when people cash their checks and a lot less other times. For that slice of the application, serverless makes a lot of sense. It's a really natural paradigm, but that doesn't mean you want to build the entire online banking application in serverless. Using them together I think is the next round of this maturity, is how do things like Kubernetes and functions come together.

[0:55:23.3] JM: Okay. I know we're slowly running up against time. I want to ask you a few more questions about being CTO of a cloud platform, because I don't get to ask these every day. There's so much blue ocean as a cloud provider. This is a very new computing paradigm relative to how big of a shift it is. There are so many managed services that you could potentially offer. How do you identify the managed services that are worth pursuing?

[0:55:54.5] JMG: It's a great trick of business, isn't it? Like how do you build the right thing at the right time? Some of it I think is — I think there's kind of two sources that we use. One is we're deeply involved in the technology space and have a long heritage in IT and from that comes our own knowledge and perspective about where people are in their journey and what technologies to adopt. We use that to inform us. If you look at containers, which is something I spent a lot of time on. This has been something I've been working on for three or four years, but it's really only been in the last 12 months or so that it's starting to kind of cross the line into a widespread adoption, and so timing that is really important.

The other thing is like we run a lot of applications for a lot of companies and we have deep relationships and we use that conversation and I personally spend a pretty high percentage of my time. 25%, 30% of my time talking directly to clients and helping them with the evolution and adoption of technology, and that of course informs me about where are they really. It's dangerous as a CTO, you can get caught up in the technology and think people are away farther ahead than they really are in their adoption. So that there's a grounding in kind of talking to clients and understanding where they are that we use to try to inform when we work out things.

[0:57:18.3] JM: Interesting. We've done shows about managing datacenters and cyber liability engineering and lots of operational challenges. As the CTO of a cloud platform, all of the difficult problems that can't be solved by other people who are maybe closer to the problems themselves, those problems eventually bubble up to you. That's kind of what a CTO does. What's the hardest challenge that you've had in your time managing a cloud platform?

[0:57:51.0] JMG: I think the hardest challenge especially in a comprehensive platform like ours is actually not a discreet technical challenge. It's getting people on the same page about how to move forward together architecturally. Getting a hundred services to agree on simple things, like we're all going to build on Kubernetes, and maturing a platform to meet the workloads needs of a diverse set of services.

I think my role as a CTO is twofold. One is setting technical direction, helping navigate the forest of all of these projects and activities and solutions that are out there. A big part of it too is building consensus on the right way for us to move forward together collectively, and that has a big impact on our customers. It sounds like an internally focused thing, but it directly results in the user experience that customers have when they adopt a platform like IBM cloud. If our stuff all works on a common singular architecture, if we solve problems in a consistent way, like we're all going to do logging and monitoring in the same way, then that manifests itself in user experience, right?

To me, that's actually the biggest challenges. One of the lessons of my personal career in software is that the software part is the easy part. The people part is the hard part. As a CTO, you have to do both, you have to solve the technology and help the people understand why that's the path they should follow.

[0:59:15.9] JM: Jason McGee, thanks for coming on Software Engineering Daily.

[0:59:18.6] JMG: Thank you, Jeff.

[END OF INTERVIEW]

[0:59:21.7] JM: When your application is failing on a user's device, how do you find out about that failure? Raygun let you see every problem in your software and how to fix it. Raygun brings together crash reporting, real user monitoring, user tracking and deployment tracking. See every error and crash affecting your users right now. Monitor your deployments to make sure that a release is not impacting users in new unexpected ways, and track your users through your application to identify the bad experiences that they are having.

Go to softwareengineeringdaily.com/raygun and get a free 14-day trial to try out Raygun and find the errors that are occurring in your applications today. Raygun is used by Microsoft, Slack and Unity to monitor their customer-facing software. Go to softwareengineeringdaily.com/raygun and try it out for yourself.

[END]