EPISODE 660

[INTRODUCTION]

[0:00:00.6] JM: Back in 2014, platform as a service was becoming an increasingly popular idea. The idea of platform as a service was to sit on top of infrastructure as a service provider like Azure, AWS, or Google Cloud. That would simplify some of the complexity of these infrastructure providers. Heroku had built a successful business from the idea of platform as a service and there was a wildly held desire in the developer community to have an open source Heroku. Deis was one of the first project to use Kubernetes as a tool to build a platform as a service and the team that was working on Deis got very early exposure to the process of building a platform as a service on top of Kubernetes.

Michelle Noorali was one of the engineers on the Deis team. When Deis got acquired by Microsoft, Michelle was working on Helm, a package manager for distributed systems that works with Kubernetes. Helm allows developers to deploy distributed applications on top of Kubernetes more easily. A few examples of distributed applications that can be deployed using Helm are Kafka, Prometheus, and IPFS.

One reason that Helm is so useful is that distributed systems are notoriously hard to configure and run and Helm simplifies that. Since joining Microsoft, Michelle has continued to work on Helm, she’s also a member of the Kubernetes Steering Committee and the Board of the CNCF. Michelle joins the show to talk about her early experiences building platform as a service and her perspective on the Kubernetes ecosystem.

It was a great discussion of the contemporary subjects of the Kubernetes world. Full disclosure, Microsoft, where Michelle works, is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:01:59.3] JM: Accenture is hiring software engineers and architects, skilled in modern Cloud Native tech. If you're looking for a job, check out softwareengineeringdaily.com/Accenture. Working with over 90% of the Fortune 100 companies, Accenture is creating innovative, cutting edge applications for the cloud. They are the number one integrator for Amazon Web Services, Microsoft, Azure, Google Cloud platform, and more.

Accenture innovators come from diverse backgrounds and cultures and they work together to solve client's most challenging problems. Accenture is committed to developing talent, empowering you with leading edge technology, and providing exceptional support to shape your future and work with a global collective that's shaping the future of technology.

Accenture's Technology Ccademy, established with MIT, is just one example of how they will equip you with the latest tech skills. That's why they've been recognized on Fortune 100's best companies to work for list for 10 consecutive years. Grow your career while continuously learning, creating, and applying new cloud solutions now.

Apply for a job today by going to softwareengineeringdaily.com/Accenture.

[INTERVIEW]

[0:03:30.7] JM: Michelle Noorali, you are a software engineer at Microsoft. Welcome to Software Engineering Daily.

[0:03:35.2] MN: Hi, thank you so much for having me.

[0:03:37.0] JM: So today we're going to be talking about containers and Kubernetes and some contemporary subjects. But I want to get some historical context because before you were working on containers and Kubernetes, you were working at Engine Yard and this was a platform is a service back in 2014 and I'm sure this stuff feels so dated but I can remember back in 2014, the early days of platform as a service, what was that like back in 2014 when you were building a platform as a service?

[0:04:09.3] MN: That was an exciting time. I actually moved to san Francisco from Georgia to take this opportunity. I wanted to focus on, back in engineering versus full stack development and are so excited to be there because we were doing all these really cool, neat things with

AWS, with the cloud and helping people deploy and manage their applications on AWS without having to deal with the behemoth that is AWS or really, any other cloud provider.

Yeah, we were — platform as a service had become really popular, kind of like 2011 around that time and it was really great especially for startups that didn't have their own DevOps teams. It's the companies like Groupon and Hotel Tonight and other big names take advantage of a PaaS to deploy and manage their applications as they were scaling without having to figure out everything in the cloud and we were so excited to kind of be a part of that journey with a bunch of startups.

It was fun, it was a good time helping solve people's problems.

[0:05:12.6] JM: What were the fundamental lessons you learned about building platform as a service or just building infrastructure technology for developers?

[0:05:22.2] MN: Yeah, I think the cloud is really still hard to figure out because you just kind of go in and you have so many things to choose from and at the end of the day, a developer is just trying to deploy and run their application, they're trying to figure out what happened when stuff goes wrong. But really, you know, just get their business logic out there so that at the end of the day, the business can keep going and growing and make money.

So figuring out like what exactly the workflow is and what things from all of the options the developer needs, that's still – is still really hard and that's something I learned while I was at Engine Yard and still continue to realize today.

[0:06:09.1] JM: Yeah, it's funny because I think developers still have those problems; the same problems of deploying your app, of like figuring out what database to use. It almost feels like we're still frozen in time, to some extent.

[0:06:24.3] MN: Yeah, I know. But I have hope! I have hope we're getting better because we're kind of rebuilding everything a little bit from the ground up, you know? A cloud became really popular, it became really, a lot more possible to do things like prototypes and get it out there and scale startups and stuff and a bunch of other things as well. But now, we realize through all the hardships of building these PaaS's, also that something like Kubernetes and having that generic abstraction layer to really deploy your containers applications, it has a lot of value and I think that that's something especially people who have worked on infrastructure tooling in that PaaS space really understand intimately.

[0:07:08.9] JM: Well, there was a lot of churn in figuring out what the abstraction layer that people would centralize around would be. I think that was in this kind of 2011 to 2017 timeframe where there were things like open stack and Deis, which Engine Yard where you worked eventually acquired Deis. Deis was this company that – I think it started, well the project was an open source project and it was also a company built around that open source project and it was a framework for doing — it was like, I think the tag line was "open source Heroku" initially and then over time, it adopted Kubernetes and started using Kubernetes as some of the underpinnings of Deis.

Can you tell me about Deis and where did Deis come along in the evolution of these different platform is a service, open source abstractions?

[0:08:07.4] MN: Yeah. So I'll tell the story because I think it's funny. I came to san Francisco, bright eyed and eager and joined Engine Yard and at Engine Yard, you know, I was actually kind of there when they plateaued a bit. Engine Yard was really popular, really great for deploying these Ruby in PHP and Node.js apps and that has gotten a little bit easier over time and so I was kind of there during a plateau period and two weeks after I joined, we got this mysterious meeting invite and we went upstairs into this meeting room and this guy walked out, his name is [Bo Rollic] and he was like, "Hello. I am your new CEO," and nobody really knew about that, it was like a surprise to most of us.

That was really interesting because, you know, him coming in presented a lot of opportunities for new change in the company and what Engine Yard realized was that containers, which were so popular at the time, they were all the — Docker was all the rage. Everybody wanted to use Docker, they just didn't know exactly how but they were so desperate to – we were all really so desperate to solve the "it works on my machine" problem. We knew that containers were going to change the game and here, you know, came along Deis and Deis had a platform as a service, open source project that was built entirely with containers and so there was this – the tag line was, yeah, “Heroku on your own machines” because you could do the get pushed style workflow that Heroku had and push, deploy and manage your containerized applications. Under the hood, they were using [inaudible] to do kind of the things that we now use Kubernetes for, which is like scheduling your containers and managing state and things like that.

So Deis was really on the ball, really an earlier player and that’s because one of the cofounders also, in the CTO his name was Gabe Monroy, CTO of Engine Yard after they were acquired. He was also an early contributor to Docker and kind of had already gotten on board and so, we thought that that would be a really great addition to Engine Yard and I remember him doing his first presentation to the engineering team and I was like, “Wow, this is so cool, this is totally going to change the game when it comes to PaaS and deploying your applications.” That happened.

[0:10:36.5] JM: When you look back, was there anything specific about Deis’s architecture that — why wasn’t Deis the Kubernetes, the platform as a service to rule them all? I guess, Kubernetes is a little bit lower level. Yeah, tell me about those, the differences in abstraction layer between Deis at that point and what Kubernetes became?

[0:10:57.4] MN: Yeah, I think Deis was a lot more high level, like you mentioned. Kubernetes is a lower level thing and so actually, after Deis was acquired, we were comparing Kubernetes and Mesos because Deis at the time was really great for what it did but, you know, under a lot of scale, it kind of broke every now and then. So we were looking to re-platform on something that was really strong, had a strong foundation and Kubernetes and Mesos — we needed some container orchestration system basically.

Container — and so when you think about container orchestrators at the time, Mesos actually a bigger name at the time than Kubernetes was. Kubernetes was a lot more nascent. Kind of looked at that and said, “Okay, this can be a core component of this Deis like workflow that we’re trying to build,” which was the Git push style workflow on your own machines but now it would be you know, with Kubernetes under the hood as well.

[0:11:58.0] JM: Engine Yard acquired Deis, Deis eventually decided to use Kubernetes under the hood instead of Mesos and then, after that acquisition, you were just working on retooling

Deis? Or what was your role? Help me frame this in time. What was your role like after that acquisition?

[0:12:18.7] MN: Yeah, basically, the acquisition happened and a few weeks later, we were told that we were going to pivot but this was like a real pivot, you know? People kind of joke about startups pivoting all the time. This was a hard pivot, we were all told like, “Just stop what you are doing.” Most of us were Rails developers at the time and they told us to learn Go. They were like, “Just make yourselves expert in Docker. Learn whatever Kubernetes is,” and at the time, nobody really knew what Kubernetes was. We barely knew what a pod was. So they were just like, “Okay, just go. You're going to help build Deis workflow,” which was like the Deis V2 container PaaS on Kubernetes.

Our role was just to figure it out and in that process, we realized there were some other gaps that could help make our lives easier as people who were building basically what was an application, a complex application to be deployed on Kubernetes. I mean, Deis workflow in reality, you know, a PaaS is just an application. We were building this like complex application with multiple components that needed to be deployed on Kubernetes so that at the end of the day, you can have that simple workflow to deploy your app.

We were early Kubernetes developers and that’s kind of where Helm came about. We were just trying to solve problems that we were facing as early users of the tool.

[0:13:44.9] JM: Now, around that time, the market for platform as a service was mostly startups. It was startups and hackers who wanted to deploy their app quickly, to move quickly. Today, I think everybody wants some form of platform as a service or some of the features of a platform as a service. You know, you have Kubernetes underpinning a lot of these platform as a service that are being offered to large enterprises; the oil companies and the CPG companies and the insurance companies that all want to migrate to Kubernetes.

So as the customer base has changed, what has changed in the tooling of what people want out of a platform as a service?

[0:14:31.1] MN: That's a really good question, by the way. I think one of the things that I've been seeing is that people want some opinions but they also really want flexibility to go around those opinions. I talk to companies who are just like, "We want an easy solution for our developer's large companies. We want an easy solution for our developers, but we also want our developers to be able to modify Kubernetes components if they need pretty easily and just kind of go around whatever workflow we've laid out."

[0:15:04.2] JM: Now, Deis had built some tools that allowed people to onboard and use Kubernetes more quickly and efficiently, but were you using these tools mostly internally to just manage containers for these hackers and the startups that we're standing up platform as a service application back in the day? Or were you thinking of these more as tools that would allow people to setup their entire Kubernetes cluster into interface directly with Kubernetes?

Did the customers back in 2014, did they want to interact? Or I guess, this is about 2015, 2016. Did they want to actually interact with Kubernetes or did they just want the abstraction of deploying an application?

[0:15:44.0] MN: I think people didn't really know what they wanted at the time. I think it's just like, the problems that we were facing at the time like you know, they wanted consistent environments so containers were really important for that. They wanted to be able to deploy those containers on their machines so you need a container orchestrator. I think people didn't really know exactly what they wanted and I don't think that there was an option either. You couldn't not touch these underlying pieces because there was no solution for being hands off.

We actually tried to build one early on like right when we acquired Deis, we tried to build kind of one of those hands off solutions where you could just deploy something. It was Deis on our platform, not on Kubernetes and it was like, "Here, just one click to player container," and that didn't really take off at the time because I don't think that people really – I think it was really early first of all and people didn't really know what they needed or what they wanted and how to wire things up.

Yeah, I don't know. I don't' really know if we know exactly what we want now either. Some people who don't really want to touch Kubernetes or some people who think that it's really important to understand all of the components. I personally hope to see a day where you don't really need to know anything about Kubernetes but you understand there's something under the hood that's scheduling your containers and figuring it all out for you.

[0:17:10.2] JM: Well, the palette of options is certainly developing quickly. Now you have manage your own Kubernetes cluster. You could use a managed service provider, you could use like Azure container service or Azure Kubernetes service. You could use the standalone containers like the Azure container instances or far gate from Amazon or you can use serverless functions, which are containers under the hood.

Have you seen any patterns or consolidation around this or are people just totally experimenting with all the things?

[0:17:41.8] MN: I think that its' so interesting because a lot of the stuff that's coming out, all the newer stuff that's coming out is like not uncommon problems. I think we're just taking problems and trying to build generic solutions around them. The managed Kubernetes services made a lot of sense, right? Because you don't want — it's really hard to manage all your, like upgrade Kubernetes versions like get it up and running and you have to have a whole team to do that and there's a lot of intricacies. So if somebody can manage that for you, you should let them do that.

At least I do. I don't want to stand at my own Kubernetes clusters, I used to do that. We had this [inaudible] up script back in the day, I don't even know if it's still around? But you'd have to like run that and then a lot of the times it would break at no one's fault, it's all really early stuff. It would break and then you talk to like, spend like three days figuring out what was happening. So yeah, I don't want anybody to feel that pain anymore.

I think we're still developing. There's still a lot that we're trying to figure out. I think what's really important to keep in mind is that – or what I really actually love seeing about the space is that everyone is just now learning the questions that they actually need to ask to be able to in the end run reliable, like large scale applications in the cloud or on bare metal or in hybrid environments. So we are going to see a lot more patterns come as we go forward.

[0:19:21.2] JM: Can you say more about that? What is being learned right now?

[0:19:25.5] MN: Okay, for example, service meshes are really new. But we didn't really – the whole – one of the main goals of the service mesh is to ensure reliable communication between your services. We didn't really know that that was a thing that you needed, right? We didn't know that there was an entire abstraction layer that you could have for that until people like from Lyft and Twitter kind of came out and said, "Hey, this is a thing that we're doing." Then everybody was like, "Yeah, that makes a lot of sense. We should kind of gather around and figure out that solution."

So that's one example of an abstraction that kind of come out. The serverless containers, the ACI, Fargate, that makes a lot of sense is like, "Okay, I just want one container. I don't need all of it, I don't need everything, I don't need a whole entire cluster, I just want one container, I just really want to spin up really quickly to do this one specifically highly, specific highly focused task, how do I do that?" And so that's an option for that. We didn't know that was a thing that we needed until later on so yeah, we're still learning.

[SPONSOR MESSAGE]

[0:20:40.6] JM: Leap.ai matches you with high quality career opportunities. You are more than just your skills and a job description and a resume, these things can't fully capture who you are. Leap.ai looks beyond these details to attempt to match you with just the right opportunities. You can see it for yourself at Leap.ai/sedaily.

Searching for a job is frustrating and Leap tries to reduce the job search from an endless amount of hours, days, weeks, to as little as 30 seconds trying to get you matched to a job instantly by signing up based on your interests, your skills, and your passions. Leap works with top companies in the Bay Area. Unicorns and baby unicorns, just to name a few, Zoom, Uber, Pony.ai, Cloudera, Malwarebytes, and Evernote.

With Leap, you are guaranteed high response and interview rates. You can save time by getting direct referrals and guaranteed interviews through Leap.ai. Get matched to jobs based on your interests, our skills, and your passions instantly when you sign up. Try it today by going to

Leap.ai/sedaily. You would also support Software Engineering Daily while looking for a job. Go to Leap.ai/sedaily. Thank you to Leap.ai.

[INTERVIEW CONTINUED]

[0:22:19.7] JM: I think another element that you might be intimately familiar with is the notion of distributed systems package management. I mean, many of the applications that you want to deploy to a Kubernetes cluster are a distributed system. Kafka, you know, it's several different nodes, how can you easily deploy a Kafka cluster? How can you easily just deploy a replicated MySQL cluster? Well, you could use a package manager like Helm and that's one thing that you're working on.

Would you classify Helm as one of these things where it's like, once you identified the issue of distributed systems, packaged management and distribution, it was another one of these, "Oh, we need to build a lot of tooling around this."

[0:23:05.7] MN: Yeah, definitely. You know, we ran into this problem and were like, "Hey, I have this application and it has a lot of components that work together and I really need to deploy this and I need to save, like I need a way to share this thing once I've figured out how to deploy it with my team and then I may want to deploy it in different environments, I may want to deploy it on my own machine or on a cloud provider. How do I configure it slightly just to, you know, do one or the other?"

So that's kind of the questions that we're asked when Helm was first being built and it was the problem that Helm was built to solve. Yeah, I think that deploying, we came up with the package format of a chart. I don't know if you're familiar? It basically holds your Kubernetes manifests which is how you define what things you want to run in Kubernetes and so basically, it's like, "I want to run this container and here's all the configuration around it," and then you can template those manifest files, those instructions to work slightly differently in different environments if you want to do that and you can package it up and you can share it with your team. That's kind of the main, like the essence of what Helm is.

[0:24:28.3] JM: That was developed when you were at Deis, right?

[0:24:30.8] MN: Yeah, that's correct. It came out of a hackathon actually when we were first starting to work on Deis workflow.

[0:24:36.7] JM: What was the impetus for building it? Was there some set of problems that you were encountering on a regular basis that led to the development of Helm?

[0:24:44.5] MN: Yeah, it was, "How do I consistently deploy something on Kubernetes and how do I – once I figured out how to deploy this thing on Kubernetes, how do I share what I have with other people?" We were finding that with Deis workflow, which was a set of components that we were running on Kubernetes. So we were like, "Okay, I made some changes to this one component and I, you know, now figured out how to run this the right way in Kubernetes. I'm going to package it up and I'm going to pass it over to you and then you can figure out your part and add to that and we can version that package so we know, you know, that this set of configuration works in one way and then the next version will do something else." So yeah.

[0:25:26.7] JM: Yeah, I guess it hadn't ever been built before because in the past, I guess the closest equivalent would be if you go into one of these cloud market places and you – the cloud marketplaces do have some templates for installing distributed systems but they're cloud specific whereas this is Kubernetes specific so you could deploy it on any Kubernetes cluster regardless of where it's running.

[0:25:51.8] MN: Exactly, yeah. We modeled this a lot after Homebrew, and Debian, and apt, yum, those types of tools that allow you to install something really simply on your operating system. We were like, "Well, instead of having one machine, maybe we can think of your Kubernetes cluster as like a distributed operating system and figure out how we can install applications simply on that as well." So it's very heavily modeled after existing package managers.

[0:26:25.3] JM: Yeah, some of the differences I can imagine, if you're developing a distributed application installation system versus one that's just for single node or you've got I guess dependency management becomes a little bit more difficult. You've got the potential for nodes to fail at any given time.

Tell me more about the differences between developing a distributed systems package manager versus a single node system?

[0:26:53.4] MN: Yeah. I think Kubernetes actually takes care of a lot of the complexity under the hood for us. So nodes failing, you know, things going wrong. You know, you basically define what you want in Kubernetes, you define maybe you have a Kubernetes resource like a pod or a DaemonSet or something and that's basically encapsulation of the application and how it runs in Kubernetes and you say, "Kubernetes make this happen." You know, you're just like giving in to the system and it is responsible for making it happen.

You know, creating Helm, we really got to take a lot of that for granted and what we focused on was mainly, "How do I describe the set of things that I want to install in my Kubernetes cluster and how do I version that? How do I package that, how do I template it so that I can configure it to run in AWS but also in Google and also in Azure and also in bare-metal? You know, how do I do those kinds of things?"

So I think the main idea here is, how do I give you all of the goodness of Kubernetes but also let you isolate things that make your application run differently in different environments and how can we package that easily?

[0:28:19.1] JM: Is that a big use case for Helm, the ability to specify different configurations for different cloud providers? Like if you want to have a multi-cloud Kubernetes cluster or you want to have a Kubernetes cluster that could be multi cloud in the case of fail over or burst capacity or something?

[0:28:39.6] MN: Yeah, that's definitely not an uncommon use case. I think what's even more common though and more easily relatable, I think is, "How do I deploy this in my staging environment, how do I deploy this in my production environment, how do I deploy this in a testing or development environment?" So maybe like that's more relatable and more common.

[0:29:00.3] JM: So Helm has these three core concepts and we talked a little bit about this stuff in the episode of Ralph Squillace, but I would like to revisit it with you because you're very involved in it.

So there is the chart, which is a recipe that describes the metadata and the resource definitions. You've got the values, which are user supplied configuration. You have a release, which is a chart together with the values you supply. Can you say a bit more about the core concepts of Helm?

[0:29:25.9] MN: Yeah, and you did a great job summarizing so I will just add to that. Yes, so when you go to deploy a thing on Kubernetes. Kubernetes is just a system that manages your containers on your actual machines. So it is going to schedule the containers for you and make sure they are running. Kubernetes has its own concepts or its own abstractions. For example, one of those concepts is a Kubernetes pod.

A Kubernetes pod would be an example of a resource in Kubernetes, or a Kubernetes resource and the thing the pod does is that it holds your application container. Sometimes a pod will hold more than one container like maybe you will have another container that deals with logging and monitoring, for example. But let's just focus on that pod holding one container, which contains your application.

So you might have an application that you want to deploy so you'll define a pod in Kubernetes. So we write Kubernetes manifests for those, which is like a yaml file or something else and then what happens is that to deploy an application, especially a complex application, you may not just have to define one pod. You may also have to define other types of Kubernetes resources like a DaemonSet is a type of Kubernetes resource that ensures that your application container is running, there is one instance of your application running on every single node or every single machine.

And so there is another concept called the Kubernetes service and that is a load balancer. So it manages the traffic to your application and so when you go and actually write up a definition for your application and how it should run in Kubernetes it ends up being multiple different

Kubernetes resources or Kubernetes components. You have to define multiple of these abstractions.

So a chart has a few different areas to focus on but the biggest piece of a chart, which is just a directory of files is the templates directory and that contains all of the manifest that you need to run your application in Kubernetes and then outside of that, you also have a chart.yaml file, which describes the package you're trying to deploy. So a set of components in Kubernetes and then you also have a values file. So you touched on that, which is the user supplied configuration.

So you can actually template your manifest with Helm. We use Go Templating and you can param try some aspect of your manifest or any aspect really of your manifest. So for example, a very common one is you would template your image name or image tag and in the values file, in the chart you would supply defaults for all of those parameters that you have defined in your templates but when you actually go and deploy your chart, if you want to override that values file, you can supply a different file with a different configuration.

So say you have something different or a staging your environment you might want to pass in some other values file when you go install your chart. That's chart and metadata, and the values file and the release is actually the thing you get when you install an instance of your chart in your Kubernetes cluster. So if you do a Helm install a chart, you get some randomly generated name for a release. So it will be something like "cute puppy". You can override that obviously but we like to have fun and I especially really like the randomly generated themes.

So you will just get some randomly generated name for release and you'll refer to that installation as a release in Kubernetes and that release contains all of the information for the Kubernetes resources you just installed. So it will say, "You have these pods, these Kubernetes service, this config map, this whatever other things." And when you go and upgrade or you can go ahead and upgrade that release or rollback or delete that release and what those things do is it applies actions to all of the resources that were defined in your release. If you do a Helm install of the same chart, then you'll get a different instance of that chart and you will have a different release and a different release name. Does that makes things a little more clear?

[0:33:59.7] JM: It does, yes. So in practice, as I understand, people use Helm to install things across their Kubernetes cluster. So things like Prometheus or Influx DB or a service mesh, for example. What else do people use in their day to day operation for Helm? Or what aspects of help do they use in their day to day operations?

[0:34:25.3] MN: I think you hit on the big one, right? So they use it mainly to install and manage those sense of Kubernetes components that are related, which make up your application. They may also use it to find charts. There is like a search command so you can search for what charts are out there. You can also do more chart management type of actions, like music to create a chart repository or some aspects of a chart repository. A chart repository is a thing that — it can be any webserver that holds charts.

But it has at the root of it an index file and so you can use the Helm CLI to generate that index file, for example and you can also use Helm to scaffold a new chart. So the Helm create command, you just say Helm create, pass in some name, we'll say "My Chart" and they'll generate this basic chart for you and then you can use that to even draw in your templates or your Kubernetes manifest and get going from there.

[0:35:33.0] JM: When I spoke with Ralph, we talked about not just Helm but also Draft and Brigade, are you also involved in those projects?

[0:35:42.4] MN: Yeah I am. Draft was a big part of my day to day for a long time and I still spend some time there. We've been working on a newer version of Helm so I have been focusing on Helm to these days. But yeah, Draft is one of the very interesting – more interesting projects I have worked on. So it is a tool that helps you create and iterate on Cloud Native applications or applications that are going to be deployed to Kubernetes.

But it does all of the things that you need to deploy your application kind of under the hood for you, which is really nice. So the back story here is everybody I worked with had basically the same set of make files that we would use to do all of the things to go from source code to things running in Kubernetes and that included, you know you had to, after you wrote your source code for an application you would containerized that, you'd write a docker file, you'd use docker build to build that application or build that container, excuse me. You'd deploy it to a container registry of your choosing. You'd then create a Helm chart and then you'd reference that image in the Helm chart and then you'd deploy the Helm chart to your Kubernetes cluster and it was just like a set of steps that you kept repeating.

So what Draft did is basically automate that workflow for us and I really like using it actually and so you have, if you're in a source code repository, you can do a Draft create and it will detect the type of application you're in. Sometimes this doesn't work all the way so there is a way to manually override this, but you can run Draft create, which will detect what type of application you're in. It will give you a docker file, a base docker file to work with. It will give you a base Helm chart to work with and then you can do a Draft up, which will build and push that image to a container registry and then it will reference it in the Helm chart and deploy that Helm chart for you into your Kubernetes cluster, which is pretty cool.

And then you can do things like, I think it is Draft connect and that is a tool that allows you to basically you get back a local host URL that you can use to then play with your app that is running in say remote Kubernetes cluster and so in that fashion you can iterate on your app too.

[0:38:18.6] JM: Very useful. So the Helm2, the updates to Helm that you are working on, what are the improvements that you are focused on? What are you building?

[0:38:28.2] MN: Yeah, so I think the biggest thing to take away from Helm2 is we are removing the server side component. So Helm has two pieces. Helm is the CLI and then Tiller is the thing that we install in your Kubernetes cluster, which then creates the releases, talks to the Kubernetes API to build, create those objects that you pass in and then manages your release in your cluster. So a lot of that can be done client side and so we are re-architecting Helm to be a single client side tool that does all of the things that Helm does for you.

So that is one big thing, and then lately I have been re-architecting the Helm test command. So Helm test is a command that you can use to run tests for your chart. In your chart you can define some test that will ensure that your chart is working the way that you want it to work and so this is something that I worked really hard on during Helm2 trying to make it a feature but not backwards incompatible.

And so, it feels a little bit raw still and so Helm3, because let's say that is a breaking change, it is like a major version release, we can make big breaking changes and so I am redesigning what the user experience would be for Helm test and then how we wait to architect it internally as well.

[0:40:06.0] JM: Is there anything particularly difficult about that refactoring of Helm particularly removing the server set components sounds difficult?

[0:40:14.5] MN: Yeah, it is a huge chunk of work. One of the people on our team, Adam Reese has been working on that and it is something that you know, we have that development branch up for people to see if they want but we are actually not accepting contributions to Helm3 at the moment because all the stuff that goes into refactoring for removing the service side pieces is really complex and Adam is doing an amazing job. Sometimes we jump in and try to pair with him but he's been really leading that effort.

[SPONSOR BREAK]

[0:40:57.7] JM: Cloud computing can get expensive. If you are 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.

This assessment will show you how you can save money on your cloud and DoIT International is offering it 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 and if you don't know whether or not you are spending $10,000 if your company is that big, there is a good chance you are spending $10,000. So maybe go ask somebody else in the finance department.

DoIT International is a company that is 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 they complete cost optimization platform for Google Cloud and that is available at reoptimize.io as a free service if you want to check out what DoIT International is capable of building. DoIT International are experts in cloud cost optimization and if you are 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 CONTINUED]

[0:43:21.8] JM: I want to take a step back because you are involved in the Kubernetes Steering Committee, you are also on the board of the CNCF; tell me what the state of Kubernetes is from that high level? When you're looking at it from the perspective of seeing adoption and different governance conversations, tell me how you feel that Kubernetes has changed in the last year.

[0:43:47.3] MN: I would say it has gotten a lot more mature, we are really focusing on putting the community first and trying to be more transparent in the way we do things and really trying to basically figure out what has been going on thus far and how we can make sure that the projects and the community grows in a healthy and sustainable fashion. So I was elected to the Kubernetes steering committee last year and it is about 13 people and we've been focusing on a few different things.

So how do we even just manage the GitHub org like the Kubernetes GitHub org? Because there is just so many people involved and there is a lot of automation in place and so we are trying to figure out what makes sense to do from a GitHub management perspective and so a lot of this is like very tedious tasks. How do we figure out what we're doing already and then make that a process and then actually go and implement it. So that is one piece, GitHub management.

The second piece is, we have a special interest groups who basically take an area of the code base and just run with it and they just specialize in particular areas. So SIG apps is the one that I was really involved in and that deals with the app's API and the ecosystem tools. There is a lot of other ones as well, SIG cluster life cycles, SIG node service kind of log. I could keep going. There are some for the cloud providers, etcetera.

So each has its own area that they focus on but up until now there has not been a really consistent way for managing those special interest groups or even having transparency into how it functions, how decisions are made, how they take accountability, etcetera and so we've been working on kind of qualifying that in what we call SIG charters and so that is also a huge process that is going on right now and then the third big area that I have been focusing on actually is the code of conduct committee.

We have a code of conduct and historically code of conducts have been unenforceable and so this committee is responsible for maintaining and modifying, iterating on the code of conduct that we have but also enforcing that code of conduct in the community and kind of making decisions and executing on those decisions, etcetera and I am just so excited about that committee because it is fences of people who just have this class of background as five members.

All of them have backgrounds and dealing with code of conduct issues in various different communities and they are bringing their expertise in, they are doing research with other groups like the Mozilla Diversity and Inclusion team and they've been doing research with other people as well just on how to create this healthy environment, how to resolve issues in the community. So we've been doing — those are the three big areas. We have been doing other things like figuring out our values and other things as well.

[0:46:59.3] JM: What are the discussions that have come up in the code of conduct area? Would that have been difficult to figure out or contentious?

[0:47:07.3] MN: I think it is not so much that the code of conduct is contentious. It is more of like it is kind of light and I think it will be helpful for us to define what the scope of the code of conduct is, where it's enforceable, where it's not. What are the norms of how our community functions because a lot of the times what happens is like, you have people from different backgrounds, different areas of interest and experiences and not everybody knows exactly what the norm is and how to work together. And so if we just define that in a document then everybody kind of has a single source of truth and I think that will be really helpful and really looking forward to that as well.

[0:47:56.5] JM: Definitely and so the Kubernetes Steering Committee that you are a part of, what does that role consist of? Is that like just figuring out the direction of the project?

[0:48:06.1] MN: Yeah, so it is basically making sure that our – the Kubernetes project is healthy and growing and if there are any issues that we're there to solve it. But a lot of the times really what we end up doing is figuring out, like we don't want to make decisions. We want to give people, like empower people to make decisions and so then the question becomes like, "Who are you empowering? Like who should make what decision?" And then that is what we are trying to find out with SIG charters and putting some processes in place because a lot of the times it wasn't necessarily so clear what direction the community should go in. So when there was like a big decision that needs to be made, process wise or otherwise, you can go to the steering committee. There wasn't that before.

Before, it was like maybe one or two people who were everybody was bombarding to get the answers to their questions or to make a decision and that doesn't scale with a project that is the size of Kubernetes and growing at the velocity that Kubernetes is and so the Steering Committee is there to make help decisions or help guide people to other people who should make decisions.

[0:49:23.5] JM: Yeah, that makes a lot of sense. One thing I wanted to get your perspective on was the business side of Kubernetes and the developments for the cloud providers, the opportunities for smaller infrastructure companies, I had a conversation a while ago with Brendan Burns that's stuck with me where he talked about the standardization on Kubernetes and that since you have a standardized layer you might see a rise in new types of business models where for example people can sell binaries that run on Kubernetes and those binaries could be deployed to any cloud provider, which I thought was an interesting idea. But you don't even have to go to that kind of extreme of an idea like a proprietary binary to see the big impact that Kubernetes is going to have. You can just talk in terms of Kubernetes being a standard deployment model for different companies that want to be multi cloud for example.

But tell me what ideas you have around how Kubernetes is just going to change the business of software engineering companies.

[0:50:30.0] MN: Yeah that's such and interesting and intriguing question as well. You know, Kubernetes just like really levels of playing field because you have this layer of obstruction that can run anywhere and you can have hybrid environments and then there is also things like IOT and edge computing and all of that. So there is just so much room to play in. I don't know exactly how things will go, but I do know that there is a lot of opportunity. I mean like I have been in the workflow space in particular, so I can speak to that.

Like the opportunity for PaaS style workflows, the opportunity for smaller infrastructure companies to create like easy experiences not only for web developers but we haven't really touched like the machine learning – I mean we have touched the machine learning space, but there is so much room to grow. It is going to be way easier for people to do research style things on a research-style workflows using Kubernetes.

It will be easier to develop different types of applications and deploy different types of applications. So I think it is not like how you deploy the application but I think the market is going to be more open in terms of what kinds of applications you deploy on Kubernetes and so there is so much space to build different types of workflows and then monetize on that. I definitely see that as an opportunity. The binary thing that Brendon was talking about, that also sound really interesting.

I have particularly seen people in the monitoring and logging space really be able to have a value add there and then in turn monetize. People like Datadog and Sysdig have just been like been on the balls since containers where super in fashion. So there is a lot of room here and then the service mesh angle and serverless angle, there is a lot of opportunity I think.

[0:52:40.2] JM: Yeah, definitely. It is going to be interesting to see which of these ideas are the big hits, because I think even the ones that are a little niche right now will find their way and they will have some kind of business model. But certainly, there will be some big hits. The edge computing side of things is, that one is curious because do you have a sense for how eagerly "edge computing" is being adopted by the giant companies that can deploy really big budgets to it? Are they still in the evaluation phase there?

[0:53:11.8] MN: I would say we are still early on but it is something that I keep hearing more and more about and so the enthusiasm is definitely building. I know for Microsoft that is something that we keep talking about or Satya Nadella keeps talking about in his keynotes and stuff. So really looking forward to how that paradigm shift evolves.

[0:53:32.7] JM: It is inevitable. I mean the idea of having like machine learning models deployed to the security systems on an oil refinery or a security drone that is monitoring the oil refinery or the ship yard or any of these other edge areas. But the time horizon is really ambiguous.

[0:53:54.4] MN: Yeah and you know I saw like "edge" being talked about in a commercial. I think it was an AT&T commercial or something but they have this really good commercial and they kept throwing the term edge out and I was like, "Wow, they are just really on the ball." So I mean if it is showing up in random commercials, yeah like you said, it is inevitable.

[0:54:17.3] JM: I guess we're at the end of our time. We obviously kind of jumped over the phase of being acquired where you were at, Deis you got acquired by Microsoft and you know now you've been working on Kubernetes tooling and obviously governance at Microsoft. I guess that switched to focusing on containers and Go when you were at Deis, that is pretty fortuitous.

[0:54:40.6] MN: Wow it was. I look back and I was so grateful for the opportunity because not everybody gets to experience that and I just really was — I got lucky. I was in the right place at the right time and it has been super fun.

[0:54:53.9] JM: Yeah, do you have any advice for people who are just getting into the Kubernetes community?

[0:54:59.4] MN: Absolutely; don't be too intimidated. I think that things will get easier as time progresses and you know this is a really open and inviting and welcoming community and if it isn't you should let me know. But you know we aim to be really inclusive. So get on the Slack channel, ask questions. It is really complex, not everybody needs to know every piece of it and just try stuff out. Don't be too intimidated is my advice.

[0:55:29.7] JM: Michelle Noorali, thank you for coming on Software Engineering Daily. It's been great talking.

[0:55:32.7] MN: Thank you. Thank you so much for having me.

[END OF INTERVIEW]

[0:55:37.7] JM: If you are building a product for software engineers, or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an e-mail jeff@softwareengineeringdaily.com if you're interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers, because I talk to them all the time. I hear from CTOs, CEOs, directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves.

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail, jeff@softwareengineeringdaily.com. If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company. So send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]