EPISODE 305

[INTRODUCTION]

[0:00:00.3] JM: Infrastructure is a term that can mean many different things, your physical computer, the datacenter on your Amazon EC2 cluster, the virtualization layer, the container layer, on and on. We refer to all of these things as infrastructure.

In today's episode, podcasters Chris Wahl and Ethan Banks discuss the past, present, and future of infrastructure with me. Ethan and Chris host Datanauts, a podcast about infrastructure. In each episode of Datanauts, the show goes deep on the topics such as networking, server list, OpenStack. I think there was as show about DevOps, or NoOps recently.

As someone who hosts a podcast that goes into similar wide ranges of topics, I find it really entertaining and educational to hear their points of view on a regular basis, because they often cover some of the same topics that I cover and express very different viewpoints, because they have very different sets of expertise. They have a much deeper level of domain expertise in infrastructure.

So if you're interested in these kinds of topics, I'd really recommend checking out the Datanauts podcast. If you like Software Engineering Daily, you might like Datanauts, and if you like Datanauts, you will love this episode of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:01:28.3] JM: Dice helps you easily manage your tech career by offering a wide range of job opportunities and tools to help you advance your career. Visit Dice and Support Software Engineering Daily at dice.com/sedaily and check out the new Dice Careers mobile app. This user-friendly app gives you access to new opportunities in new ways. Not only can you browse thousands of tech jobs, but you can now discover what your skills are worth with the Dice Careers Market Value Calculator.

If you're wondering what's next, Dice's brand new career pathing tool helps you understand which roles you can transition to based on your job title, location, and skill set. Dice even identifies gaps in your experience and suggests the skills that you'll need to make a switch. Don't just look for a job, manage your tech career with Dice. Visit the app store and download the Dice Careers app on Android or IOS.

To learn more and support Software Engineering Daily, go to dice.com/sedaily. Thanks to Dice for being a sponsor of Software Engineering daily. We really appreciate it.

[INTERVIEW]

[0:02:42.3] JM: Chris Wahl and Ethan Banks are the hosts of the Datanauts Podcast. Chris and Ethan, welcome to Software Engineering Daily.

[0:02:48.7] EB: Hey, nice to be here.

[0:02:49.3] CW: Hey! I'm waving my hands, actually.

[0:02:53.3] JM: Great. You're the hosts of the Datanauts Podcast, which is all about infrastructure. What defines that term infrastructure today, because we've got so many different things that could potentially be referred to as infrastructure?

[0:03:09.5] EB: Infrastructure from our perspective is really anything you would find in a datacenter. For example, my background is network engineering, so that would be any of the network physical infrastructure that's there, and virtual infrastructure. Then, add to that, servers, and storage, and security, and all of that mix put together, the physical stuff that gets racked and plugged together that is serving some purpose in the datacenter, to my mind is infrastructure. Chris, feel free to add or expand.

[0:03:37.9] CW: I thought all you like was networking stuff, Ethan?

[0:03:39.8] EB: Maybe.

[0:03:42.2] CW: Yeah, I don't know. I tend to think of it as everything, that it's the physical, and the virtual, and the software that goes in between. It's less software development. It's not building the software, but I think everything else is kind of fair game; the config management software, the hypervisors, all the managers of managers, all the way down to the actual hardware. Although it feels like — I don't know Ethan. I think we've talked about the hardware stuff less and less nowadays, unless it's bleeding edge stuff on the MIDI industry or something like that.

[0:04:10.5] JM: I want to take sort of a historical perspective, because you guys have both been in the industry for a while. Ethan and I were talking before the show started, you guys met at a Cisco Conference a long time ago. If we'd look back over like a 20-year time horizon, I think that Amazon web services might have been the most important thing to happen to what we call infrastructure. Maybe you agree with that, maybe not, but the thing I'm curious about is what was the world like when AWS was created, because I started my career in software like after AWS was created. Maybe you could contrast the worlds of before and after AWS.

[0:04:48.4] CW: I never thought of it that way.

[0:04:51.0] EB: Yeah.

[0:04:51.0] CW: I'll make the comment that now you make me feel really old in infrastructure, because I started the 90s. It was more than just that, it was also — There was virtualization, but not as we know it today with x86. It was really a very slow moving — I don't know. Very detail-oriented job, I felt like, because I was a system administrator for a very long time and I felt like most of my job was just knowing how to do a little bit of everything. It's kind of the pre-siloed egg, because I was the one adman working for General Motors for a long time, and then a couple of other places, and it was just like, "How do you set this group policy object, and how do you physically bootstrap a Windows server, or a Linux box?" I don't know. It was easier and the same time slower, the golden days.

[0:05:34.7] EB: A lot of really rolling custom infrastructures, some hardware and software combination that was very particular to a business' needs and a lot of them were — A lot of enterprises were similar, but a lot of enterprises were — Almost all enterprise ended up being a

snowflake of some kind or another. I think if you want to compare that to Amazon and just the cloud model generally, you're seeing applications need to comfort to a standard of predictable set of parameters that make them easier to manage no matter what environment you put them in. It didn't used to be that way. It was very much a roll your own and kind of go with your gut and your own instinct and experience. That really put an imprint on what the infrastructure was like for a given organization.

[0:06:20.3] JM: The approach of having snowflake servers used to be totally acceptable and there's nothing wrong with that. Today, it's more like if your server is not something that can be dispensed off due to a network frits or just a machine crashing, then basically you're doing it wrong, assuming you're on the cloud.

[0:06:42.4] CW: To a degree, ye. We had to plan for those things — Back in the day, we had to plan for those things, drive failures were common, servers would get corrupt or something would happen. They had a way to restore those services, it was just slower and done by hand. It was like you hand the blueprints to build the artisanal chair again, you have the chisels and everything, it just took a while.

Whereas now, it's expected to be, I guess, A; automated, like it can fail and something else will build it, not a person. B; not impactful, whereas if your production primary file server puked back in the day, that would be like a huge outage. It would be all hands on deck to rebuild that thing and get it back online; redo the shares, permissions, et cetera. Go grab the tape from Iron Mountain, restore it. It was just a lot of work. There was no other way though. There really wasn't a good alternative. We didn't have config management software. Building an instance of a server required a lot of handwork. You could script some of it, but it just wasn't very friendly to a lot of automation. There wasn't a lot of tools, unless you rolled them yourself by writing batch scripts and PowerShell exists, so you had to write everything in batch, or actually dig into like C code or whatever. The plethora of open source tools and the community that exist today to show examples of that just didn't, it didn't exist at all.

[0:08:00.9] JM: In retrospect, when you look at the rise of AWS and its significance, why do you think it took a long time for there to be other major cloud service providers that got their clouds going?

[0:08:18.2] EB: it's expensive, and it's a big investment, the standup of public cloud and then to get people to begin moving their workloads into that cloud, because moving into that cloud is not always an easy thing. You still get enterprises today that can see the value prop of AWS, or whoever the public cloud provider is. It doesn't make it easy for them to pick up their apps and just move. It doesn't even mean it's the right decision for them to make depending on a whole lot of things about their business or what the business requirements are.

You get AWS going up with the cash and the capital to standup that infrastructure and begin attracting people. You've got a young group of new developers eager to try that and startup companies who are able to launch a business without having to invest in their own hardware, datacenter space, and that really spun up Amazon quickly as a business. Chasing them and competing them against them is hard. We've seen some public clouds very recently here fail, just unable to compete effectively enough to make it work. Cisco has shut down their inter-cloud service, HPE with their Helion cloud is down now. We'll see what other attrition comes. It's a tough business to be in.

[0:09:34.2] CW: Plus, why would you want to give up that control? Go back to 2005, or whatever, all the way to 2010. I don't know. Maybe the years are a little off, but if you're the IT ops team, the cloud offered you nothing you wanted, "I have no visibility in the infrastructure. I don't know what the heck it is. Someone else is running it. It's expensive," because you're being paid by the drip. You already have all the infrastructure there anyways.

I don't know. It just wasn't a very attractive value proposition at the beginning, and it took a lot of education, I think, before people understood what it could be used for and how it could be used for. Originally, it was like, "Everything in the cloud, move it all there," and we're standing around like, "That's stupid. I have five petabytes of stuff here that's mandate to be governed, and I have to know where the data is, and I have to have control," and all these things that were considered like fluffy and not considered by those that aren't in on operations, like, "Who cares?"

There had to be a point where we both had what we needed, because there was a shift where everyone is just doing the black ops type development in the cloud and IT doesn't know about it, and then I would stumble upon so many of these horribly designed cloud environments,

because no one with any kind of chops around networking, and security, and access control, and roll-based access control, or anything had any idea what they were doing. They were just building stuff and having fun, and then it goes in production and it's a mess, and it's vulnerable. At the same time [inaudible] wasn't scalable like that. It's tough.

[0:11:05.5] JM: That sounds like me, I would be the developer who's like having fun with stuff and I'm just like deploying it and then it's just a disaster when it goes to production. Maybe you could touch on the roots of this DevOps thing that we're moving towards that we've had for a while now. When I started as a software engineer, there was already this idea of DevOps. As I understand, from doing shows about this topic, there is a history of conflicts between dev and ops and I'm curious how that relationship between the developers and the operations people. How has that evolved from the time that you've seen pre-cloud through post-cloud?

[0:11:53.6] CW: My experience — The most recent poor experience is 2011 before I moved in the consulting full-time, and that was where I was in charge of all ops, and it was a two man team for thousands of servers. It's very, very light, very highly virtualized. We had somewhere around 500 developers, all offshore, all contract, writing code. There was a lot of rules around how the code could actually get into production. They obviously couldn't push it to production. Ops was supposed to deploy and manage and it was this quarterly cadence of disasters where you'd spend all weekend putting it in there.

There was just a lot of bad blood, a lot of, "We know this is going to be crap, because the system itself was broken." No, I don't think developers want to write crappy code. I just think they're not given the requirements clearly and they don't know what's going on, because they're hidden from production. We never talked to them face to face, like I didn't know any of them, because there was the product and project managers were in the middle.

The whole model is broken, but a lot of the hatred was between ops and dev, because we never got to sit down and talk about what the overarching design is. All of the incentives were misaligned. They were incentive to just ship quickly and hit these little boxes and there was no incentive to kill bugs, make the deployment easy, keep the service online. To me, that was the major issue, and it's tough moving forward to fix that, because there's a lot of emotion there, there's a lot of bad process.

I was in a company of 17,000 people. You can't easily fix that, or even communicate that you want to fix it very easily. Those were the major problems.

Moving forward, I think what we're really seeing is that the development community has a lot of great ideas that ops can take advantage of around doing commits to infrastructure as code, actually collaborating on what's going on, having peer reviews. There's a lot of great workflows that ops can take advantage of to make their life better.

From where I sit, a lot of the DevOps concepts and thought leadership is really just around how do we treat infrastructure more like developers treat their code, and that's been a huge improvement for me beyond just the whole let's collaborate between dev and ops. There's my two cents. I guess I'll let Ethan fill his part.

[0:14:11.7] EB: The big thing that I would highlight, Chris, that was very much my experience as well, was there is this wall that prevented communication between dev groups and ops groups. Development team would come, new release, gotta get deployed and you just kind of get this ticket as an ops person saying, "Yeah, we need to make a hole in the infrastructure that the dev can fill us up with." "Okay." No requirements, no nothing given, and then if things aren't working right or if it's not performing right, all of a sudden there's a big crisis meeting and you're in a conference room on the hook with all these sour looking people and everyone's pointing fingers at everyone else. You finally pull together the right brains to sort through it from both dev and ops and figure it out and it finally gets to be fine, but it never needed to be that way had the work been done on a collaborative format upfront.

I think that is part of what DevOps is about. It's really an expectation that dev is going to understand operations and infrastructure a bit and vice versa, that operations folks are going to have some clue about the applications and how they work and what the demands on their infrastructure are going to be. It's really breaking that wall down and making a two way street of communication such that you can avoid these problems and have much smoother success and production and rollout is much easier.

Really, the only way you can get to this fast refresh cycle kind of world where we just patch this and we're going to release that, and it's an everyday or every week, or a couple of weeks sort of release cycles instead of the monumental once a quarter release cycle as if you've got a good full of communications and dev has the ability in an automated way to deploy code to infrastructure and so on.

That's really behind a lot of these. Part of it is just to make thing better, and part of it is business necessity to move along as quickly as some companies want to move along with their apps. You need to have that in place these days.

[SPONSOR BREAK]

[0:16:11.2] JM: You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, Glucode, lots of iteration, and lots of frustration.

The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today, that's exaptive.com/sedaily.

The Exaptive Studio provides a visual environment for using back-end, algorithmic, and front-end components. Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3.

Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better. Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW CONTINUED]

[0:33:54.9] JM: When I think about the ways that the developers have gained more responsibility over the infrastructure, I think about things like the infrastructure's code, movement, or technologies like Chef and Puppet, and then there's also OpenStack, which is this infrastructure as a service, open source software where you could layer it on a cloud, or you could layer it on your on-premise machines, and then we had Cloud Foundry which built an open source platform as a service. Then, we've just developed more and more stuff. There's, obviously, Docke, which has — That's often associated with being important to DevOps, which is a containerization tool that lets you slice up your operating systems even further than virtualization did.

I know one of you mentioned virtualization before cloud as being another fundamental turning point in what the idea of infrastructure is. Docker disrupted how every application developer works. Then, more recently of Kubernetes, we'd get into that. From your perspective, how did the infrastructure story, how did the idea of infrastructure changed with Docker, or did it just feel like nothing new? Did it just feel like yet another slicing up abstraction thing?

[0:19:02.7] CW: At least with Docker, we are — I would say, as an infrastructure person, I already had kind of a concept of what was going on because of x86 virtualization. It prepared me more than it had when I first encountered virtualization from VMware from a hypervisor perspective, which was around 2006 or so. That was a big mind-leap to try to get over the idea of running different servers on the same physical server. If you think about it, that's tough. It's a tough leap to make. It was a little bit easier to just say, "Well, it's kind of like that, but we're doing it more at the kernel level within a server," it can be virtualized or physical. That sort of technology had worked in previous implementations that I've done before with Solaris Zones.

It was less of a shock, but more of a — The thing that I felt was a lot of snark around it because everyone is like, "Oh! This is going to solve the world. All development will be amazing and wonderful now." I'm like, "Well, it doesn't solve security. Networking is still a problem. It still has to run somewhere. Artifacts had to be set like, "Okay. It's another way that you can slice things up and present it," but I didn't see it as the nirvana that was going to save the world. It's just yet another way that you can build and present applications.

[0:20:12.1] JM: Docker, of course, gave rise to Kubernetes, which is the container orchestration engine from Google. Kubernetes is yet another way to manage infrastructure, it's much like OpenStack was in some ways, or like AWS was, is somewhat like Cloud Foundry in a sense. I get the sense that there is this different level of affinity with Kubernetes. People seem to love Kubernetes — I don't know. The community seems really strong —

[0:20:43.7] CW: It's because they have Kelsey Hightower, man. Woo!

[0:20:45.6] JM: Because they have Kelsey Hightower. Honestly, that is like obviously a component of it, because he's such an effective evangelist. Is there something special about Kubernetes when you look at it relative to the timeline of when the these different infrastructure management, the big infrastructure management tools have come out. is there something special about it or is it just the fact that Google is putting a lot of muscle behind it. Does Kubernetes feel different to you?

[0:21:13.8] EB: I think we got to back up a step and talk about OpenStack first and see what some of OpenStack's challenges are. OpenStack was going to be your private cloud platform. That's what everybody was going to run. It was the thing that was going to change the world and save the world and all the vendors with their infrastructure were bragging about how, "We are integrated with OpenStack, and here's how," and demos were all about making their hardware work within an OpenStack context.

What's happened there? It seems to be difficult to use. Still, there are companies that have made their money making OpenStack easy to use through either their integration services or their own flavor of OpenStack that they've released.

As we move ahead, OpenStack development slowed down a bit. There's still some stability challenges. There's even talk of using a scheduling tool like Kubernetes to keep OpenStack running, because some of the back-end services have a tendency to fall over. OpenStack's development does seem to be, again, slowing a bit, maybe it's going to stabilize and get better.

A lot of folks just like — We try to make this work and it's not working. Kubernetes comes along from Google. They release it to the world and go, "Here, look at this thing we made. Do you like

it?" It seemed to be still a little challenging to use and it's still is. It's got a lot of rough edges on it, but it's more stable, seems to integrate with Docker very well. Not that it's limited to dealing with container workloads from Docker. It can do other workload management as well. It's got some great features that people like, auto scaling, Achieve One, integration with cloud services. A lot of people looking at that going, "Hmm, this does a lot for me." I think that groundswell of popularity behind it is coming from other shops that are also gloaming in to it and putting weight behind it. There's strong development initiative there too.

Is it another one of these fads that, "Well, it's sexy today and everybody loves Kubernetes," but eventually the shine is going to wear off and people look to something and goes, "I don't know." It certainly feels differently with Kubernetes to me than a lot of these other tools.

OpenStack was the new shiny for a while and the shine definitely wore off and it does not seem to be coming back. There are shops that are using it, it's not like it's disappeared from the planet by any means. Kubernetes is fulfilling a lot of — They're not the same tool exactly, but certainly fulfilling a lot of the promise of OpenStack was and people seem happy to push that development cycle along and bring some maturity to the product.

[0:23:44.7] CW: I just think Kubernetes, at least what it's supposed to be is what I feel that the datacenter is supposed to be. This kind of glob of compute that accepts workloads and figures out where to run them based on policy and the exposure of resources underneath the hardware, and that's it. Ideally, with the gold standard of what we're trying to do, I'm not saying Kubernetes is like amazing at it and is always going to be number one, but that's the architecture that — If I were still an IT practitioner in a customer environment, that would get me excited, like, "Yes! I want to learn this. I want to run it. I want to contribute to the community and figure out how to take advantage of this, because this is really what we've been trying to do for a very long time."

It's just been driven by failed vendor products for very long time to build these private cloud environment with these — Take DynamicOps and the old B Cloud director stuff that went into your datacenter. The vRealize Suite I think is the most current iteration. Originally, why DynamicOps came out as a way to build private clouds — It was a C# application that it had to install on Windows. It had a big database dependency. It's like this monolithic app that you

would use to supposedly build distributed applications in a cloud environment. It was very like — I felt the hook pulling against my cheek. I'm like, "Where is the line?" It's sucking me in on this.

There's been a lot of that. We've been kind of cloud abused from a private perspective, and so the idea of something like Kubernetes where it can work on prim, it can work across clouds, and it's really meeting what I want for my environment. It's exciting, so I've been kind of following it for that reason.

[0:25:16.9] JM: If I understand the history correctly, OpenStack came out when Amazon was getting going and people were like, "We need an open source version of AWS," and then Rackspace sort of put its muscle behind OpenStack and said, "Hey, we're going to run our cloud using OpenStack so we have this aligned interest with the open source community." What was different about that scenario than relative to the scenario today where Google says, "Hey, we're going to start using Kubernetes internally and we're also open sourcing this thing and we're running our cloud this way." What's the difference?

[0:25:55.7] CW: There's much fewer fingers on the pie. Google can use their own internal software development methodologies to build something versus — Originally, it was NASA and Rackspace with Nova and a couple of neutron. I think it was the old name for that networking component. They were using something that worked internally. Google hadn't released anything at that point that I'm aware of.

The two of them collaborated, build something and then more and more people keep getting added to it, now there's like three billion people that are all platinum, and gold, and silver level stewards of the project. To be honest, I don't really follow OpenStack as closely as I used to three or four years ago, versus Google, where it's like, "Hey, we write software for a living. We need these things." What is it like? Invention is born out of necessity. Therefore, we need something that can do this, because it's a problem very unique to our environment.

[0:26:43.9] JM: What I also hear you saying is that there was too much bizarre with OpenStack and not enough cathedral.

[0:26:49.0] CW: You try to be everything to everyone, you're going to be nothing to everyone, or nothing to no one. Yeah, there was a point where I felt like there was 20 something projects in the core Open — I'm exaggerating a little bit. It felt like driving up to a sonic and having the menu of everything you could possibly want. You can't do hotdog, steak, stew, and sandwiches good. You got to pick two, right? It was like the fast food of cloud where they were trying to go everywhere.

[0:27:17.0] JM: The Cheesecake Factory.

[0:27:17.8] CW: I don't admit to going there. I've probably seen it on television. Yeah, you got to have some focus. You got to be able to say no, and that's tough when everyone has an equal vote and there's all sorts of misaligned interest. Cisco's going to want to go one direction, VMware's want to go a different direction, Rackspace's want to go maybe a third direction. Everyone has their self-interest at heart, because they're all kind of making bets on what this thing can do. Whereas versus Google where it's like, "Yeah, we used this for a long time. It's great. We've kind of iterated upon it, so we're going to release the old version, which his Kubernetes, essentially." Now they're evangelizing it, 'because — I don't know. It's a great idea. People like it. I think it helps solve real problems, versus OpenStack, which — I'm not trying to — It sounds like I'm just bashing the hell out of it, but a lot of great people put in a lot of time.

[0:28:05.5] JM: Oh yeah. I did a show on OpenStack. It is very interesting. I hope people check it out. It's just older. Every older project relative to a newer project is going to have its pluses and minuses. What I think is so interesting about cloud, and it's like at every point in cloud, it's like, "Okay. We're finally getting a hang of this cloud thing. I think we have a good handle on it," and then like five years later, things look significantly different just due to trends and new technologies coming out.

I think in another two to five years, we're going to be looking back and saying, "Wow! We were doing a lot of management of our cloud that we don't have to do anymore. Obviously, the buzz word that articulates this is serverless, which is the idea that you have some code and it execute against a cloud on some arbitrary server and you don't know what server it is that you just get it kind of opportunistically, and the cost structure of sererless movement is much cheaper than kind of spending up servers and managing them.

Then, there's also these things like if you think about Google's cloud API, it's a cloud vision API, just make an API request to get an objective identified. What is in this picture? Then, manage machine learning and things like firebase that really take care of your scaling up and down. It really does feel like we're moving in a direction where we're not going to be managing servers as much as we are managing code, brcause we kind of like — There's no service where you don't want scale. There's no service where you don't want elasticity. Do you agree with that, or does that sound too strongly ideological?

[0:29:55.4] EB: The cost model you mentioned around serverless, I think I read — I don't know if it was an Amazon quote or what, but the suggestion was if you're running against LAMDA with function calls, you're getting an 80% savings as supposed to actually running some full server instance upon EC2 of something.

Right, there's cost savings there and then a lot of what's already being abstracted away for you in public cloud is abstracted away even more, because now you're just running functions against you don't even know what. It's just — It's there as the servers that you provisioned. You are even more distant from the physical underpinnings that make up that cloud. If there's good cost drivers to do it and you could build your app for it, woohoo! It sounds like a good thing. You still got storage to manage. You still got security concerns. You've still got a lot of resiliency redundancy performance concerns that all matter as your application begins to scale.

In a sense, right, we're not managing infrastructure from a certain point of view, but form another point of view, there's still a very heavy ops component that comes into this. Your application has got to be performant or you your user-base, however big that user-base grows, it's got to have availability for wherever your users are. There's still operational things that are going to have to be thought about whether you're running on infrastructure as a service or whether you're running more a platform as a service, or whatever you want to call serverless.

[0:31:21.0] JM: I don't think we ever get to where we don't manage infrastructure. It's just maybe some of the terminology changes, but the discipline is going to be very much the same what operations people do and how they think about it and design infrastructure. All those

principles still apply even though — Again, there might not be some virtual server you're pointing to and say, "My code runs there." You still got to think about these things.

[0:31:44.6] CW: To me it's not big of a difference from the transitions that I've made with virtualization, and even some of the CICD pipeline tools that I use. As an example, the VMware host that I run, they're stateless. I don't care about them. I treat them as a pool of resources, which is kind of what we're doing with some of the functions as a service and serverless. I don't know — There's some pool of servers running underneath there. I don't really care or know about them. That's one kind of similarity.

The other one is I use Adfair to do unit testing for all of my power show builds, and so I just throw code at an end point and it comes back and says yes or no to test passing. There's already examples in the infrastructure world where we've taken that step. They are as fully baked as something like functions as a service, or serverless R. it's certainly not going from zero to a hundred. There's also been steps taken in that world.

I would agree mostly with what Ethan is stating there, that probably a lot of the names and such change, so there was going to be some service somewhere, because you don't want to have stuff manager environment running in the thing that's managing the environment, vCenter server is an example. I never let the vCenter hosts, the VMware hosts that are running vCenter, the management components just live in general population. I always kind of put that aside. I run all my really critical stuff that I don't want a dependency chain to exist within.

I'm not saying that's going to be on prim necessarily, but I feel like we're always going to have some small population of endpoints or servers that are put aside — Maybe we're talking like 3, or 5, or 10, or something like that. That's just outside the scope, because you want to have to troubleshoot a thing that's broken and have that also cause you not to be able to do the troubleshooting. That's just like a, "Hey, let's not do crazy things here. We've learned these lessons over the last 30 years, don't just them throw them away."

[0:33:33.5] JM: We've been talking mostly about the software components of infrastructure, which we would expect on a show with Software Engineering in the name. I want to talk a little about like life inside of a datacenter, because as I understand you both have little experience in

that, and I have zero experience. I've done no shows about it. People who want to know more should probably tune in to Datanauts and the other, Packet Pushers Podcast. Tell me what life is like inside of a datacenter, because I have no idea.

[0:34:06.6] CW: Very cold and very noisy. That's pretty much the answer.

[0:34:12.8] EB: Life, like, "Boy —." Go ahead Chris.

[0:34:15.7] CW: I'll say there's a delineation. The life of the datacenter engineer and then there's actual time spent in the datacenter, which is pretty negligible these days. Unless you're adding physical kit to the environment and rack-stack that kind of jazz, you're not going in the datacenter. Gone are the days where you actually work at the datacenter. Mine was usually 30-minute drive or an hour drive away. Which is kind of a transitory thing, right? I remember when I would in there all the time with crash carts, checking out a server and doing something physical and this is really before IPMI and remote KVM was a thing that you could afford.

Mostly the work though is spent on making sure that all that infrastructure is online, it's responding within the correct amount of time. Originally, that was up-down type test, ping test, et cetera. Now, it's looking at user experience type, dashboards, and making sure that capacities are within — Have you ever seen like the old movies where they're at the nuclear silo and there's all these little gas cages everywhere and they just don't want it to go red, "Just don't red." That's pretty much what it is. As long as everything is green, we can go and play Starcraft or something and —

[0:35:20.6] JM: It's Homer Simpson in the nuclear facility.

[0:35:23.9] CW: Yeah. I don't know if I'd like to draw a parallel with IT practitioners and Home Simpson, that feels a little [depricational], but it is a little like looking at those dashboards, yes.

[0:35:35.7] JM: That's not what I meant at all, I just meant like he's just sitting there and he does nothing until stuff gets red, starts flashing.

[0:35:43.1] CW: Yeah, because there's so many things moving around and doing things, and a lot of that you're — I don't mean this in a bad way, you're ignorant of what they're doing. I know that this particular volume or LAN has two terabytes associated with it and I don't want it to go above 85%, but I don't at a glance know everything that's going on with that, and that's okay, because until it triggers an alarm, I don't really care. That's kind of the background of systems administration that work operations, whatever, the knock, if you want to call it that. The rest of the time is spent fulfilling the needs of the business and the greater IT ops with help desk if you're at the lower level, all the way up through architecture and engineering discussions at the higher levels of the organization.

[SPONSOR BREAK]

[0:36:29.5] JM: Simplify continuous delivery with GoCD, the on-premise, open source, continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines, and you can visualize them end to end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.io/sedaily, you can find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent predictable deliveries. Visit gocd.io/sedaily to learn more about GoCD.

Commercial support and enterprise add-ons, including disaster recovery, are available. Thank you to GoCD and thank you to ThoughtWorks. I'm a huge fan of ThoughtWorks and their products including GoCD, and we're fans of continuous delivery. Check out gocd.io.sedaily.

[INTERVIEW CONTINUED]

[0:37:41.2] EB: I'll categorize it a little differently than Chris.

[0:37:43.5] CW: Because you're the network guy. You just draw packets off the floor, right?

[0:37:46.8] EB: Yeah, a lot of that. My experience has really been — In organizations, there are times of growth and there are times what I would call run and maintain. Life in a datacenter,

when you're in a period of growth. Growth could be defined by a business that's growing and so you're adding capacity, or growth could be — It's a budget refresh and we've got to get some of the old stuff out and bring some of the new stuff in. There's major projects and initiatives to improve the IT infrastructure in that way.

These times when you're working very hard on bringing new infrastructure online, or transitioning from old infrastructure to new, or both in a lot of cases. There's a physical component to that. You're racking equipment, you're cabling that equipment. You're bringing power into that equipment. You're making sure that the equipment doesn't exceed the weight capacity of the floor in that particular part of the center.

[0:38:35.5] CW: Oh, that's a good one.

[0:38:36.2] EB: All of these sorts of things. These projects tend to go on for a few months. It's a long bit of time to get new infrastructure online and being used. Particularly, if you're moving into it, the old infrastructures got to have the data that's on and get moved into the new infrastructure. There's usually a big build up to the cutover day. This is the day that we're going to cutover. That modern infrastructure things are different and maybe that's not so hard to do if you've got applications that can work in more of a private cloud environment and your workloads are more portable.

In the old day, that hard cutover sort of experience and a lot of stress would go on the operations teams just getting to that point where they were the new — The new infrastructure was there and finally ready to e used. Now, finally, we did the cutover on the weekend and it's Monday morning and everyone's there early and haven't worked all weekend going, "Okay! Is it all working? Do we have any trouble tickets? Is anybody complaining? Is everything working okay?" Then, the final confidence, you can shut down the old equipment and pull it out of the racks.

There are those times in the business where there's a year or maybe a two or even three year period where there's money and the company is growing, and so you're constantly on new projects and standing up new infrastructure and working towards some goal that is going to meet business objective, and IT is going to be a big part of that.

Then, there's the run and maintain phase which is more like, "I don't want anything to break. If something breaks, I got to get right on it and get that thing fixed and squelch that red light just as quickly as possible." Touches little as possible. Work on your documentation and your procedures and cross-training people and all that kind of stuff, because you're not quite in a super high pressure scramble mode. It's a little bit more of that, "Here's what we've got and we're going to make the most of it."

Yeah, little upgrades here, and, "This server needs some more RAM," slap that in. "I need a new blade on this switch," slide it into the chassis. Easy stuff like that, but it's not that the big project mentality that takes over you and the lives of several other people on your team to bring it to life.

[0:40:42.9] CW: I will say there are some cool points where you're getting access to like a federal or a bank secured datacenter. There's three layers, and there's two mantraps, and they had to put a special sticker on the bottom of your laptop, and there's a guy with an AK escorting you to the cage. It can be kind of cool sometimes. It's not always just putting crap inside the rack and removing other crap from the rack.

[0:41:04.2] JM: I have this very brutal picture of what it's like to work inside of a datacenter, basically, because there was a show I did with one of the brothers who started DigitalOcean, Moisey Uretsky, and he had this quote, he was like, "If you haven't bled inside of a datacenter, then you don't know what it's like to start a cloud.

[0:41:22.4] CW: That is so true.

[0:41:24.5] EB: I have bled. That's true. That's funny.

[0:41:26.2] CW: There are a lot of sharp metal pieces inside of servers and your hands will all of them.

[0:41:30.9] EB: I can take you to a datacenter right now with a cabinet that's got my blood on it. No word of a life. That is funny.

[0:41:35.9] CW: Kind of fertilize the datacenter with your DNA for sure.

[0:41:41.4] JM: On that note, I've never bled writing software. I would love to know — I think most of the people listening to this podcast have like six to maybe — Well, probably, one to 10 years of experience working in software. I think they tend to be on the earlier side of their career. What are the durable concepts that a developer should know about infrastructure? What are the timeless things that they should really keep in mind, because I think a lot of developers who listen to this show, maybe they have just written some Python programs, or like they're just worked in Java. They have no idea what the concept of infrastructure even is.

[0:42:19.9] CW: I think the easiest and probably most obvious when I say it out loud is that assume it's all going to fail. I think that was the biggest issue that I —

[0:42:28.6] JM: Whatever it is.

[0:42:29.1] CW: Yeah. Having worked with infrastructure for 17 or so years, everything will fail at some point. Discs are the obvious one, fans, servers, power. I've seen the dumbest things happen in a datacenter in which things go down. I've actually seen where someone changes an SFP on a piece of gear and that has a cascading failure and actually takes on the entire datacenter.

Just assume when you're writing code that you can't make any assumptions. You can't assume this endpoint will be online, and I've seen where people have written code in a such a way that, "Well, the DNS server is always online, and if it's not, there must be a problem with my software." And it's like, "Oh no, just the DNS server went offline. It's not you." Or just don't assume that it's online. Have a secondary in a different datacenter, or check the web. Just don't make assumptions that infrastructure will be ever be online 100% of the time, because it's mathematically impossible, so just don't do it.

[0:43:20.8] JM: Okay. I know we're drawing up against time at this point, but very interesting topics we could discuss. The Datanauts Podcast is what you guys are on here kind of discuss. We haven't really talked about it, but I would love to just like hear — What are your goals with

the Datanauts Podcast? Why would people want to go and listen to that? What are the topics that you like to explore? What are the themes of the Datanauts Podcast?

[0:43:48.4] EB: The single biggest thing with the Datanauts Podcast we would describe is busting siloes. In other words, Chris and I have both operated in environments where there seem to be a big divide between dev people and then people even within operations within the infrastructure world kind of walled off from each other, "The security people did the security thing, and you can't talk to them, and the storage people did the storage thing, and you can't talk to them," and so on all down the siloes, and we wanted to bust those siloes up, because we see the datacenter and infrastructure as a unified system that delivers applications.

What better way to properly design a complicated system like that than getting all the stakeholders together, all the technology experts together, and getting those big brains around the whiteboard to put a design on the whiteboard where everyone's had a chance to provide some input and say, "Oh, if you do that, that means this for my system, and that's not really good, but we could do this other thing, and get it designed right up front." How do you do that? You foster conversations. You begin to make what the other person does and that other cubicle who you never talk to less foreign and less secretive and mysterious. Open the black box. Take a look inside and see what everybody else is doing.

Plus, it's really a reflection of something that we are seeing happen to our peers, where a lot of us have been deep specialists for a long time me networking, then very heavy into that, when early in my career, it wasn't that way. I did networking, and storage, and servers, and security, and kind of got into this networking silo.

What we're seeing now, that is kind of going back to the way things were where folks are becoming more generalist. They may have a deep specialty or favorite technology they like to work with, but by enlarge, they are becoming more generalized. Convergence, hyper-convergence is reflecting that as those systems become more and more popular. That's, again, the big goal of Datanauts is to have those conversations and make it so that the people who are deep specialists become much more familiar with what's going on in other parts of the datacenter and come out of their siloes and chat with everybody else.

We want that to be not just within ops, security, and storage, and so on, but also developers and application rollouts. The open source folks that are giving us Kubernetes and so on keep talking about what's going on out there. Have a chat with the Docker folks and see what's up with containers and their slant on things. We had a chat with CoreOS and figured out why they do things the way they do things and what their angle is, et cetera. It's a whole lot of bringing people together in the infrastructure world.

[0:46:20.9] CW: I'll just add. It's also great, because we're long time ops folks and we're learning a lot too. It's a good litmus test if we feel like we get down with an episode and we've learned some things. People in our shoes probably are too. We don't think we're anything special, we just use it as a gauge to help really just spread information and education and learn some cool stuff, because we all want to make better data.

I don't think anyone out there wants to set their datacenter in fire and make it worse, unless you've got a horrible boss or something like that. That's outside of my scope. We all want to make things better, spend more time at home and spend less time with a pager or being on call. I think these technologies are what are going to make that a reality, going to make life better across the board. For me, that's a personal journey to become better at kind of more your side of the isle. I think the podcast helps that.

[0:47:08.8] JM: Yeah. What I'll say in closing is one of the things I really liked about Datanauts that I don't get as much from doing this show is like the combination of breadth and historical context, and I know I'm probably making you feel old again saying that, but it's like super valuable to me, because I get breadth in a modern contest, and that's mostly what I do shows on. It's very nice to listen to a podcast where you guys are always well-prepared and you have a depth of knowledge and a breadth of knowledge that is really entertaining. Yeah, thanks for coming on Software Engineering Daily and I've really enjoyed the Datanauts Podcast. I hope you guys will come on again in the future.

[0:47:50.7] CW: I think that's the nicest way we've ever been called old before. Thank you.

[0:47:55.8] EB: Thank you. Jeff, folks that want to find Datanauts can go to packetpushers.net to find the Datanauts show and other shows about infrastructure, engineering, more focused on networking, but we got a lot of content there for people and it's all free to download.

[0:48:09.3] JM: Great. All right guys, thanks for coming on Software Engineering Daily.

[END OF INTERVIEW]

[0:48:16.3] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. Thanks again Symphono.

[END]