EPISODE 585

[INTRODUCTION]

[0:00:00.3] JM: The Kubernetes ecosystem consists of enterprises, vendors, open source projects and individual engineers. The Cloud Native Computing Foundation was created to balance the interests of all the different groups within the cloud native community. CNCF has similarities to the Linux Foundation and the Apache Foundation. CNCF helps to guide open source projects within the Kubernetes ecosystem, including Prometheus, Fluentd and Envoy.

With the help of the CNCF, these projects can find common ground where possible. KubeCon is a conference organised by the Cloud Native Computing Foundation. I attended the most recent KubeCon in Copenhagen. KubeCon was a remarkably well-run conference and the attendees were excited and optimistic. As much traction as Kubernetes has, it still is very early days, and it was fun to talk to people and forecast what the future might bring.

Two of the people I talked to were Chris Aniszczyk and Dan Kohn, who are the COO and Director, respectively of the CNCF. I was curious about how to scale an organization like the CNCF. In some ways, it's like scaling a government. Kubernetes is growing faster than Linux grew, and the applications of Kubernetes are as numerous as those of Linux.

Different constituencies want different things out of Kubernetes, and as those constituencies rapidly grow in number, how do you maintain diplomacy among competing interests? It's not an easy task and that diplomacy has been established by keeping in mind lessons from previous open-source projects. I enjoyed these conversations with Chris and Dan and I hope you enjoyed them too.

[SPONSOR MESSAGE]

[0:01:55.7] JM: Software Engineering Daily is brought to you by ConsenSys. Do you think blockchain technology is only used for cryptocurrency? Think again. ConsenSys develops tools and infrastructure to enable a decentralized future built on Ethereum, the most advanced blockchain development platform.

© 2018 Software Engineering Daily

ConsenSys has hundreds of web3 developers that are building decentralized applications, focusing on world-changing ideas like creating a system for self-sovereign identity, managing supply chains, developing a more efficient electricity provider and much more.

Listeners, why continue to build the internet of today when you can build the internet of the future on the blockchain? ConsenSys is actively hiring talented software developers to help build the decentralized web.

Learn more about consensus projects and open source jobs at consensys.net/sedaily. That's C-O-N-S-E-N-S-Y-S.net/sedaily. Consensys.net/sedaily. Thanks again, ConsenSys.

[INTERVIEW]

[0:03:11.5] JM: Chris Aniszczyk, you're the COO of the CNCF. Thanks for coming on Software Engineering Daily. You've been involved in a series of open source communities, most notably Linux and CNCF. How does the Kubernetes community compare to previous open source communities?

[0:03:26.8] CA: That's interesting questions. In terms of growth, we sometimes have the – we affectionately call Kubernetes, Linux and the cloud, right? If you saw some of the research we've done in CNCF regarding velocity of open source projects, Kubernetes comes fairly close to Linux in terms of speed and so on, in terms of pull requests coming in. It ranks, I think number two behind Linux and throughput.

In terms of communities, everyone uses different tools. Linux is so old-school, get format-patch mailing, like LKML is just like a bunch of patches flying through. The mailing list, Kubernetes use the more modern tools. They have stretching Github to the limits.

[0:04:10.7] JM: Which also happened with Linux? Well, they didn't even go on Github, I guess.

[0:04:14.6] CA: No. Linux is explicitly anti-Github, or at least he's quoted to be anti-Github.

[0:04:19.9] JM: Even if you wanted to be, Github wouldn't work for Linux, right?

[0:04:23.6] CA: It's a different workflow. Even the Kubernetes community has complaints around just scaling out Github for its use cases. They've even used – they built third-party tools to help deal with a review backlog. There's actually even startups out there, like reviewable.io, which is basically helping projects that deal with just – what happens when you have thousands of pull requests? How do you know what ones you've looked at, what you haven't. Reviewable.io is pretty cool. I highly recommend people check them out in terms of what they're doing.

[0:04:50.9] JM: Is that code reviews a service company?

[0:04:52.7] CA: Kind of. It connects directly to Github and it basically gives you a nice view of your backlog of reviews, and also it knows which ones you've looked at last and it tracks that, or Github doesn't – has no concept of that. It gives you a better interface for review backlog, and the Kubernetes community still uses it.

[0:05:11.5] JM: You're the COO of all this madness and the chief operating officer at any company, as the company scales the chief operating officer basically has to change their role every other week and do more things, or do different things, or delegate things. How has that changed over time?

[0:05:27.8] CA: I mean, when I started CNCF, I was the founding executive director, then we're like 28 or so members at that time, and two-and-a-half, a little over two and a half years we've grown as an organization. CNCF I think now is 23 employees covering marketing, PR, developer advocacy, technical writing, events and mix of everything. I've shifted from executive director to COO. Now I'm managing a bunch of aspects. It's all over. At the end of the day, the whole goal of CNCF is to basically support our projects.

Essentially, we're in service to our projects and whatever their needs are, like we'll do best to accommodate that. Whether it's a new website – some projects wanted security audits recently, so we're like, "Cool." We'll go figure out how to do third-party security odds to improve your security processes on so on.

We just react on the needs of our projects and members, essentially. Yeah, it's been all over. You know what's funny, I swear to myself that I would never do another startup, because before joining CNCF, I spent five years at Twitter and was there super early, where it's been only a 100 or so engineers before. When I left there was 2,000. After about five years I'm like, "I don't want to do that crazy startup experience again."

CNCF looks pretty good. Nice little not-for-profit foundation and we've grown from 30 or so members to 216 as of today in about a little over two and a half years, so it's just been bonkers in terms of membership growth and even from a staff, we went from one to about 24 in two and a half years.

[0:06:58.9] JM: There are open source projects that get really big and then have organizational issues that end up, or adoption issues, or just market issues that end up making the open source project less successful than it could have been. What are the ways that you, or what are the things that you've learned from seeing previous open source projects and how are you avoiding that problem?

[0:07:24.1] CA: When we started CNCF, we looked at other open source foundations and projects and try to take in some lessons from them of how can we actually build something that could learn from other people's mistakes and lessons. A couple things was if you look at things like the Apache Foundation arguably has lots of great projects out there, or they have something called the Apache way. Like every project is intended to follow the Apache way, share the processes, governance and so on.

At CNCF, we decided that that's limited, because each project definitely has different scope, different velocity, different levels of commitment. We went with the like, bring your own governance approach. As long as you publicly document how your decisions are made and how maintainers are elected for projects, we don't necessarily care. It was an interesting idea, but we thought that in theory if we – a lot of projects be a little more flexible, they're not going to feel bounded by what people consider – sometimes people are like, "Oh, Apache is heavy weight." We didn't want to have that mantra of there's a lot of these heavyweight processes, where most of the projects that we're taking into CNCF already were fairly functional, they just didn't maybe

have their governance defined in a way that was public and transparent and made sense, so other people could join the project.

[0:08:35.7] JM: What about testing? Because Dan was talking about testing yesterday. Do you enforce some testing policy?

[0:08:42.8] CA: The other lesson is other foundations out there, like either eclipse or OpenStack, sometimes they do annual releases like, “Hey, let's get all of our projects to release at the same time. We're all going to test them together.” We've opted not to do that. All of our projects are independently run. They define their own governance, how they run their tests and so on.

What CNCF does as a service is if you go to a CNCF.CI, we have a public system out there that will take our projects, test them on all the different cloud providers out there, both public and bare metals, so like AWS is your OpenStack, bare metal cloud providers packet on bare metal, and just run a battery of tests and see if it works. We offer that as a service, but that's not a blocking thing for projects. They are not going to not be able to do pull requests, because their tests are failing on one cloud. We do it as awareness thing. We don't want to be like a gate. We just want to provide them information to see that, “Hmm, maybe our stuff is not working on one cloud. Why is that? We should fix that, because that test is particular ready.” It's a really cool website. Go to CNCF.CI, check it out. It's a fun project.

[0:09:45.1] JM: At this point, I have a pretty good understanding of the ways in which the CNCF is similar to a startup. What are the ways in which it is very unlike a startup?

[0:09:53.5] CA: We are a non-profit, like everything gets reinvested into our projects.

[0:09:59.7] JM: Well, sure. Organizationally, organizationally.

[0:10:01.7] CA: We generally don't have – we don't necessarily have a full-blown engineering team. We're not going to do the work that we expect our members and projects to do. What we focus on instead is things that are additive and assist projects to become easier for people to contribute to, or consume. What that means is what do engineers typically don't – what they not

like to do? Writing, documentation, website work, design work. CNCF has staff that helps these for projects.

We have a team of technical writers that will go out and improve our project's documentation, the designers that help with website work and so on. We help with that type of work, that just makes it easier for projects to be consumed and use, but we don't actually have staff full-time that is working on feature work for these projects.

[0:10:52.9] JM: Makes sense.

[0:10:53.8] CA: Yeah, and also, it would just be hard for us to compete with our members for talent, which is the other type of problem.

[0:10:59.3] JM: Speaking of that, diplomacy is a huge aspect of your role. You are managing diplomacy between these giant cloud providers, or maybe putting up policies within the CNCF, such that you are not required to do diplomacy, like just saying, "We just don't do that. We do not comment on that. You pay us money as sponsors, we give you a booth at the expo," but what's the diplomacy?

[0:11:26.3] CA: I see it like, where it's like sports, right? Let's take soccer, for example, where the teams are essentially the member companies all competing with each other and we're essentially acting as referees. If someone jabs someone, we call that out and do red flag, or red card, that that is not allowed by the rules. Our job is basically to ensure that things are neutral and companies are playing by the rules, because they all signed up. There's a charter that dictates the values and rules of the organization that everyone's expected to follow and we act as essentially referees to enforce those rules amongst our members.

[0:12:07.7] JM: Has that charter had to be tweaked over time?

[0:12:10.0] CA: It's so in the formation and bootstrapping phase of the foundation. It took a long time to get done, because anytime you're trying to build consensus with a large group of people, it takes a while. We have made some small updates over time just to fix small issues. Nothing's

perfect, right? It's probably had two or three minor updates. In general, the mission, the values have stayed fairly consistent.

We are going through a fun process of updating the definition of what cloud native is, which is baked into the charter. That's been a fun, I don't want to say bike shedding process, but it takes a while. When everyone's offered their opinion and you need to come to consensus, these things sometimes take a while.

[0:12:49.2] JM: I've obviously heard the phrase that the goal right now is for Kubernetes to become boring, but boring in a good way, because Linux is boring and people know what they're going to get from it.

[0:12:58.2] CA: People use the word boring, but I think just like mature, like omething that is stable, rock solid. I don't think necessarily that technology will become boring, it's just going to become things stable and mature that's there. Linux is everywhere right and people tend to forsake that a little bit, that it's grown so much that it's literally in almost every device we use, it's in fridges, TVs, and people didn't happen overnight. It took a long time. We essentially want Kubernetes to be something similar, where it falls behind the scenes, where every cloud whether it's public or private essentially has the Kubernetes API there and Kubernetes sitting there. Essentially from an engineering point of view, like POSIX. POSIX is something people take for granted. Kubernetes essentially will become like a distributed API version of POSIX.

[0:13:45.5] JM: What happens after that? After Kubernetes is born, then it seems like there's a room for a lot of exciting stuff.

[0:13:51.5] CA: All the fun, now that we standardize on a way to do distributed development across different clouds, now we could build tools and users out there could potentially have more portable applications, because who wants to be locked into one cloud service, or provider? It's just not healthy for you. It will kill your ability to innovate, because what you want to end up seeing is all the different cloud providers and services, they're compete on an even playing field and continuing to better and improve things and you have the option to move to whichever one best suits your needs as a business. CNCF and the whole cloud native

movement in general, it wants to enable that. Essentially, we want to enable a multi-cloud world that is fair and workloads are more portable amongst.

[0:14:33.1] JM: Do you have any more futuristic projections? Because that's very exciting, I agree. I asked Brendan Burns this question, he was telling me that he saw a world in which you could have a return to this static binary that you purchase, just like you used to buy a box of software off the shelf, except this time you're buying a distributed system off the shelf and you can just go and install it on any cloud provider you want.

[0:14:55.7] CA: Yeah. I mean, that would be interesting. What interests me in future directions, I would be fascinating to see, like if we get to a point where you have this standard distributed API everywhere and you're essentially saying you could put constraints on a workload, you could be saying for this particular workload, I want this workload to be completed as fast as possible and money is no object and just go, right?

There'll be maybe a broker that goes, say like, “All right, we got this work load come in. Here's their constraints and then maybe cloud providers, or people bid on that.” Almost like high-frequency trading, right?

[0:15:30.0] JM: Sounds like Ethereum.

[0:15:33.5] CA: I want to avoid any blockchain-related topics, but high-frequency trading works. Yeah, we may move to a model where there's actually brokers where those essentially act as middlemen to compete for workloads.

[0:15:44.1] JM: Fascinating. Yeah, I think that will happen. Just a couple more questions, so is the CNCF involved in the Kubernetes conformance project?

[0:15:54.1] CA: Yes. We basically host the conformance efforts.

[0:15:58.1] JM: This is how a cloud provider builds a standard conformant form of Kubernetes, so that every cloud provider has some compatibility layer.

[0:16:06.7] CA: Correct. It's a working group under the auspices of CNCF that has the Kubernetes community involvement. We've helped basically guide the process and enforce the rules that we came up to make that happen. Essentially, it's cool. It all lives on Github, so if you go to github.com/cncf/k8s-conformance, and you go there, the tools to do run the tests are all there. All the tests that go define of what it means to be a Kubernetes certified thing are all there and documented, and everyone's open to contribute to and say like, “Oh, maybe we should add more tests right to it, or this test is a bad one, so we can improve it.”

It's all there and you could actually see all the different cloud providers and startups that are doing cool things, submit their test results in real time and see, “Oh, great.” Look, like EKS, maybe just certified the 110 version of Kubernetes. It's all public and transparent, which I think is cool.

[0:17:03.4] JM: Last question. I was doing a lot of reporting around the time of the cold container orchestration wars. That was around when you were getting involved in the OCI around that time.

[0:17:13.2] CA: I'm still Executive Director of OCI. Literally just before this, I was at an OCI face-to-face meetings.

[0:17:20.6] JM: Yeah, so what are your takeaways from that period of time, that arduous container orchestration period of time?

[0:17:26.8] CA: Oh, my gosh. Yeah, container wars, the orchestration wars, the back to container wars again with a different runtimes vying for being the de facto way Kubernetes runs workloads is just fascinating. Red Hat is investing a lot in creo, Docker as container d. Google just announced it. It's just fascinating to see as – What OCI solved in my opinion was it got everyone in the room together and said that like, “Look, this stuff has to be done in a –” we need to spec out what a container runtime is, what an image is, we don't want to compete at that level. It's super silly to compete on how containers are actually defined and laid out on disk. There should just be one way.

If you essentially use a service and you build your container, you deploy it to the cloud of your choice and then you're like, “This is not good enough for me. I want to move to another one.” For you to go back and have to rebuild everything just so you can move to another service is absolutely insane, right? OCI basically helped prevent that issue and get the whole industry onboard on specking out what essentially – kudos to Docker and a lot of stuff came from their technology and specked it out and say this is the standard.

It also help neutralize patents in play. A lot of people don't realize a lot of standards body have IPR agreements that all the members agree to saying that, “You know what? We're going to forego any patents and not switch each other in the space.” When OCI I hit 10, there is an IPR agreement that said anyone that is using an implementation of OCI forgoes the ability to sue related to patents to that specific limitation.

[0:18:54.8] JM: That’s awesome.

[0:18:55.4] CA: It neutralized things. It was a small thing that happened behind the scenes that really helped things not turn to a crazy –

[0:19:01.3] JM: I didn’t realize that side of it. Well, Chris Aniszczyk, thanks for coming on Software Engineering Daily. It would b great to talk again in the future.

[0:19:06.7] CA: Yeah, absolutely. Thanks for having me.

[0:19:07.9] JM: Okay.

[SPONSOR MESSAGE]

[0:19:17.3] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with container technologies like Docker and Kubernetes, so you can monitor your entire container cluster in real-time. See across all of your servers, containers, apps and services in one place with powerful visualizations, sophisticated alerting, distributed tracing and APM. Now Datadog has application performance monitoring for Java.

Start monitoring your microservices today with a free trial. As a bonus, Datadog will send you a free t-shirt. You can get both of those things by going to softwareengineeringdaily.com/datadog. That's softwareengineeringdaily.com/datadog.

Thank you Datadog.

[INTERVIEW]

[0:20:12.1] JM: I'm here with Dan Kohn, who's the Director of the CNCF. We're at KubeCon. Dan, welcome back to Software Engineering Daily.

[0:20:17.9] CA: Thanks. Glad to be here.

[0:20:19.8] JM: I want to start off by talking about serverless, because I've been having a lot of conversations in the hallways around serverless and how that intersects with the Kubernetes community. Why is serverless important to the Kubernetes community?

[0:20:34.6] DK: I think serverless, which is best known for AWS Lambda has gotten a ton of people excited, because it's legitimately exciting. It's a way of deploying apps that seems just easier breaking them up, things like the millisecond billing and infinite scalability. What its shown is different ways of working with computing that can just seem so much more pleasant and fast, higher velocity than what people use today.

One of the things that we've been thrilled to see and I love the fact that Kelsey did this demo at the end of the day Wednesday, he originally – this is his third KubeCon that he's co-chairing and last for a while. He originally wasn't sure he was going to do demo, but he was excited enough about this cloud events work that's come out of the CNCF serverless working group that he decided to get up and demo it.

What's been fun is to see that you can actually decompose some of those ideas that people are so excited about from AWS Lambda, and then use them in different ways on other cloud providers and also on top of Kubernetes.

[0:21:38.1] JM: One question is if you're going to build a server list-like platform on top of Kubernetes, so that anybody that deploys a Kubernetes cluster can have serverless functionality on top of their Kubernetes resource scheduling, should that serverless functionality be bundled into Kubernetes, or should it be a world of letting a thousand flowers bloom and having all these different server list projects that people could deploy to Kubernetes? Does it make sense to have an opinionated way to do serverless computing within Kubernetes?

[0:22:13.8] DK: Almost certainly not. The Kubernetes community is pretty clear about wanting to have Kubernetes do less. They're trying to take as much code out of the project as they can. There's a pretty active effort to try and strip out all the code from the different cloud providers and put those into separate repos, so that Google and IBM and each of them don't need to make changes to the core Kubernetes in order to fix provider specific bugs.

By far, the biggest strength of Kubernetes, excuse me, in my mind is the APIs are well regarded and have pretty clean interface points. It's designed as a system that you can build on top of. Now with in CNCF, I think it would make a ton of sense for some serverless projects to come in, one or more. It's not clear yet, but there's definitely, I mean, I think at least four of them are presenting here at KubeCon Cloud Native Con; Kubeless, OpenFaas, Vision and Nucleo, and I believe there's some others as well.

With the serverless working group, we created this serverless landscape and you can see it at s.cncf.io and then there's a link to the static version of it. You can see that there's a lot of different options out there. My expectations that they're going to build on top of Kubernetes, but that they would remain separate projects.

[0:23:33.3] JM: Serverless is appealing to me as somebody who reports on different engineering subjects, because there's a lot of scheduling going on when you spin up a serverless function, it's going to get scheduled against some opaque blob of computing, and you should be able to either say, “I just want that to be scheduled and I don't really care,” or you should be able to tune that schedule however you want, which is why there's a lot of subjectivity around who builds what serverless computing framework.

I think something similar could be said about machine learning, where you have these varietal workloads. I believe that there's some keynote talks this morning about machine learning. Are there going to be similar questions related to the ML scheduling side of things that are being asked around the serverless scheduling side of things?

[0:24:26.8] DK: I think if I had more time with a whiteboard, the one behind you, we could come up with some clever two-dimensional diagram. There's at one dimension at least is how long a function is designed to last. Serverless is a classic example, where I want to resize a ping. I should be able to do it in a hundredth of a second, or something. If people upload a million of them, I'd like to be able to do them all on different machines and then spin down and such.

At least one axis of that is how long it lives. I think most ML workloads tend to have a much longer cycle that they want to keep living for a long time, or at least the training piece of it needs to keep going for a longer time. Then maybe the classification piece can be much faster. I certainly would say that the aspiration of Kubernetes and the platform that we're building is that it should support all of those workflows. I mean, in a lot of ways it already does. I think there's still optimizations and tweaks and such.

In Austin, five months ago there was a pretty cool demo from a company Blackrock that was showing Kubeless; one of these offerings running on top of Kubernetes and that they were using that in production today. You can say, “Oh, why wouldn't any company just use AWS, or Google functions, or one of these others?” Blackrock doesn't want to use the public cloud for its internal data. It is pretty great that they're – now they're off obviously big enough that it's pretty – but it's pretty great that they are able to get the same functionality on their own. They're also looking at all the ML things. The idea that the same servers, the same resources can be scheduling all of this in an optimal way.

[0:26:06.7] JM: The point once again is that this is not really something that the core Kubernetes team needs to think about as much as just their basic primitives of creating resources and scheduling those resources. They should not be thinking about are we building resource scheduling capacity for serverless, or for regular enterprises, or for machine learning cutting-edge stuff? That stuff is not in their purview, right?

[0:26:36.3] DK: I mean, I think in reality when you get to any edge case, then you need to look at are there tweaks or optimizations that are necessary? The obvious one for serverless is that it does take Kubernetes a little bit of time to spin up a new container? Interestingly, AWS Lambda is also a container-based system and it takes them a little bit of time to spin up a container. They have some tricks around warm start and hot start containers, and you can do the same thing with Kubernetes, or similar things with Kubernetes.

As of this moment, I don't think there's any primitives missing. I think it is possible with the Kubernetes API, but it's quite likely that you need this custom resource descriptor, CRD, or other additional code. I mean, it's early days on this stuff, so there's just a lot of innovation going on. What's nice is that if it turns out that there is an API missing, or one that's sub-off to more stuff, there's a whole process in place that you could create patches and get that contributed over time.

[0:27:36.4] JM: That cold start, hot start problem stuff I've talked to different serverless people about this problem, is that something where if you were running a serverless function, you would want to have something in place at a lower level of Kubernetes that actually does, is aware of the fact that this is a serverless function unit you're solving for the cold star problem? Or is solving for the cold start problem a more general case that you would want to be solved for all containers that would be spun up on Kubernetes?

[0:28:07.5] DK: I think it's the latter. I mean, the simplest way to think of warm start is just, are you willing to dedicate a small amount of resources to having this function in-memory and ready to go, so that you really can get sub millisecond response time the first few times? You do it, so it's a set of trade-offs and knobs. My understanding is that today, is that all the knobs are currently there.

[0:28:29.2] JM: We saw the ballerina announcement. Ballerina is this cloud native programming language. Meta particle was announced at the last KubeCon. Why are language level abstractions so important?

[0:28:42.6] DK: I think I would probably generalize my answer and say that to an amazing degree, people now agree that Kubernetes will be the underlying platform for infrastructure

going forward. There is also an almost unanimous agreement that the current way of deploying applications onto Kubernetes is not going to be the winning one. It's not the optimal one. The idea that you need to build the container yourself, and then write some YAML and push that up; it works great and I mean, tons of folks depend on it, but it doesn't seem optimal to anyone.

What there's no agreement on of any kind is, well what should the winning technology be? In some ways, serverless is like a PaaS, a platform as a service. I think one of the reasons that Heroku and Cloud Foundry excited so many people, but didn't necessarily take over the world is that everyone wants their own PaaS, but they don't necessarily want the same PaaS as everyone else.

One way of thinking of Kubernetes and the larger ecosystem is it's this amazing toolkit for building PaaS, but it lets each enterprise or startup, or even individual customize it the way they want to. I see meta particle and Ballerina and then some others that I would put in that same category, perhaps at slightly different tiers are some work out of Azure, Draft and Brigade looking at how you develop the closed loop cycle of making a change, pushing it up. There's a project at a company called dataflow telepresence, that actually got submitted to the CNCF sandbox, and I think is likely going to be coming in. Then some work out of Google on Scaffold and Kaneko. These are, I mean, you actually can look on the interactive landscape. L.cncf.io and take a look at these under the app definition and the CICD area.

I definitely am not making any bets on what the winning technology, or winning technologies would be, but it seems like an area where there's just tons of innovation going on. Each of these teams are looking at each other and saying, “Oh, that's a clever idea. We could implement that.” The bigger picture for me is just that everyone would like to make developers lives easier, but not necessarily put a straitjacket on them. The fact that it will be Kubernetes underneath and that at the end of the day you can just build a container and upload some YAML if you want also gives you that escape hatch.

[0:31:14.7] JM: You gave a presentation earlier this week called How Good is our Code? Tell me the thesis of this presentation.

[0:31:24.5] DK: The thesis is that when you look at cloud native computing, which we define as containerization microservices and orchestration, that actually the most important aspect of that by far is a different layer of the architecture. It's continuous integration and CI. After and the talk is up on – it will be up on YouTube and you can link to it in the show notes, but the idea after doing a few amusing looks at Juicero and Burning Trucks and rabbits and some other things, is to say how good is our code not good enough? We need to have the tools and processes in place that allow us to continuously update it.

I make this point that this interactive landscape I keep referencing at the l.cncf.io is the example I use in the talk, is 40,000 lines of code. That's less than 0.1% of the total amount of code that it takes to run it. That's a huge problem, because not only can there be bugs in my own code, but all those libraries and frameworks that I depend on Linux, Kubernetes, NodeJS, Webpack, Babel and hundreds of others can have bugs in them as well. Thankfully there's lots of developers out there fixing the bugs, but I only get to take advantage of that if I have a way of continuously rebuilding, testing and then redeploying my code.

[0:32:45.0] JM: Are there specific areas of the Kubernetes codebase that you're concerned about as being more untested, or more raw?

[0:32:54.3] DK: That's an interesting question. Kubernetes has a fantastic infrastructure and I really want to credit Google here where they have invested huge amounts of resources in the continuous integration, and it's only just now starting to get transition to CNCF and some of the other providers. They've really said –

[0:33:12.4] JM: They had an internal continuous integration process?

[0:33:16.2] DK: Not really. I mean, it actually just uses Jenkins, and it runs on Google Cloud. They're running millions and millions of jobs. Every time, quick aside when you do a pull request on Kubernetes, the Kubernetes bot writes into the pull request and says, “Okay to test.” Then any one of hundreds of people can say, okay to test. Then the Kubernetes bot goes off and runs that code on CI. Today that's using Google's Cloud resources, but we are trying to transition that and have it have it go through CNCF. I mean, that's four years now that they've been supporting all that work.

As a quick aside, this is the first conference where we had a telepresence robot for people who couldn't be here that you could book 30 minutes remotely on the robot. We called it Kubernetes Bot, and there's a little tag on it that says, “Okay to test.” It's rolling around the expo floor.

[0:34:11.9] JM: The side piece of that, one of the projects that CNCF have invested a ton into and is quite excited about is software conformance. I'd say that one of the areas that we’re particularly interested in is essentially is characterized as the external API, so not just all the internal work that all the tests are doing, but there are a bunch of conformance tests and we now have 55 vendors that have certified Kubernetes implementations. What it means to do that is that you use this test harness called [inaudible 0:34:44.2] and then you run the subset of tests that are conformance tests, that test the external API.

If you're conformant with all of those, you upload the logs to a Github repo, and then we say yes, you're certified Kubernetes. Today, those tests are not as thorough as we'd like them to be. One of the things that CNCF has begun investing in this year with an external test development company based in Argentina, is to write additional tests.

OIne of the changes that we were very pleased to see that the CIG architecture group in Kubernetes made is a new requirement that you can add functionality to Kubernetes and it can get through alpha and beta, but to actually become stable it has to include these conformance tests built into it.

The way to think of this CNCF investment is we're back filling some of the technical debt for existing features that didn't have conformance test with it, but the hole won't get any deeper from here .Any new code that's contributed and gets accepted will always include those conformance tests. Hopefully we'll dig our way out of that hole before too long.

[SPONSOR MESSAGE]

[0:35:59.3] JM: Your enterprise produces lots of data, but you aren’t capturing as much as you would like. You aren’t storing it in the right place and you don’t have the proper tools to run complex queries against your data.

MapR is a converged data platform that runs across any cloud. MapR provides storage, analytics and machine learning engines. Use the MapR operational database and event streams to capture your data. Use the MapR analytics and machine learning engines to analyze your data in batch, or interactively across any cloud, on premise, or at the edge.

MapR's technology is trusted by major industries like Audi, which uses MapR to accelerate deep learning in autonomous driving applications. MapR also powers Aadhaar, the world's largest biometric database, which adds 20 million biometrics per day.

To learn more about how MapR can solve problems for your enterprise, go to softwareengineeringdaily.com/mapr to find white papers, videos and e-books. MapR can leverage the high volumes of data produced within your company. Whether you're an oil company like Anadarko, or a major FinTech provider like Kabbage, who uses MapR to automate loan risk, and has done 3 billion dollars of automated loans to date.

Go to softwareengineeringdaily.com/mapr to find out how MapR can help your business take full advantage of its data. Thanks to MapR for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:37:41.6] JM: Conformance testing for those who are unaware, meaning the Kubernetes platform that some vendor has developed is interoperable with the other Kubernetes conformant vendor-specific Kubernetes deployments.

[0:37:57.4] DK: Yeah, it connects back to the conversation we're having a second go about deployment and development platforms, where a really simple way of thinking of it is for one of these platforms, like draft or scaffolder. I mean, there's dozens of them. If they run on Vanilla Kubernetes that you download from the – and install, will they run correctly on open shift? Will they run on GKE and EKS and all these other platforms? The point of this whole conformance to test suite is to say yes, that all of these applications will be compatible with all the certified Kubernetes implementations.

[0:38:32.3] JM: When you think about that 55 vendors that have been able to adopt a conformant standard, that's a pretty amazing feat of diplomacy. That's what I think, like the CNCF is all about this these feats of diplomacy. I talked to Chris a little bit about this yesterday, and I don't think that's an accident. It seems very deliberate that as an organization, you have figured out how to maintain diplomacy among all these different organizations, these different open source projects, these developers. How have you done that, or more specifically how have you avoided the pitfalls that as humans, right, we're fallible to fighting with each other and bickering with each other and taking advantage of each other and backstabbing each other, even if we're talking about open source software, how do you avoid that?

[0:39:25.6] DK: I definitely would say that it's probably harder than it looks. There's just a lot of telephone calls, would be the simplest answer even more than e-mails, or Github issues, or anything else. I will say that it is the huge strength of having a neutral organization where CNCF is part of the Linux Foundation, we're nonprofit, there's no stock options, there's no incentives to favor one versus the other. I think what all of these vendors are looking for is to be treated reasonably.

We've had good luck thankfully on putting together a draft proposal of – so it was literally a year ago. It's an amazingly fast process as well. At KubeCon Berline, I did an early morning meeting right before my keynote and I was able to get several dozen vendors together and say, "Hey, here's the program we're thinking of launching, here's our aspirations for it, here's our process for it, anyone was welcome to participate in it and you didn't have to be a CNCF member. Then we genuinely work to take people's suggestions and improvements into account.

I don't mean to make it out as own and then we all held hands and such. I mean, there were real disagreements along the way. It's really one of the most amazing things I've ever been involved in in my career. I'm incredibly grateful to all the other participants of the working group and all the vendors that have stood up. I never imagined that it would be 55 implementations, and that we would have just the entire field, I think is the key point. There's five or six non-certified Kubernetes out there and they're actually here at the conference, and I keep bugging them. There's no lack of conformance that just haven't quite dedicated the engineering resources to finish the tests and upload them, but they all promise that they will soon.

[0:41:10.7] JM: Not to go too deep on this, but does this have something to do with, like do you think changing trends and human nature, or the way that the tech industry has adopted relative to, I don't know, oil companies, or banks? Not to knock on all companies and banks, but I just –

[0:41:26.6] DK: No, no. I don't think there's any definitely no changes in human nature. I think the biggest thing by far is just the advantage of CNCF coming later after other projects. I feel like we get to look at the mistakes that other projects and other standards efforts and such have made and learn from that. The fact that we do try to be students of history, and our aspiration is always to get to go make entirely new mistakes and not replicate those older ones.

I mean, I won't name it, but I happen to know a standards effort that had a set of profiles. One of the profiles was settings, like do you have a settings dialog box? Then a bunch of providers came up and said, “Oh, look. We’ll certify to the settings dialog box and now we can claim that we're conformant with the whole standard,” which is awful and a shame, but of course the organization could have reacted to that and it disallowed it and other sorts of things. The fact that we have seen some different forms of behavior in the past and that we were able to have both a stick and carrot of saying, “Hey, this is a huge industry.” We think certified Kubernetes is going to be an incredibly important brand in that you want to be part of it has been really helpful.

[0:42:38.8] JM: It is the carrot and the stick. It's not about the open source community as a whole having some institutional memory that well, we've made mistakes in the past that have turned into tragedies of the commons and have led to tons of technical debt and lost business opportunities is –

[0:42:53.1] DK: No, no. I think it's that aspect. Again, I don't think there's any magic that I or people at CNCF. I mean, the folks that we're dealing with tend to have the same institutional memory that we do. I mean, they're very thoughtful, engaged people. When we can say, “Hey, that proposal would lead us down this dead-end,” they tend to be extremely responsive.

[0:43:10.8] JM: That history has proven to screw us over.

[0:43:14.3] DK: Yeah. I mean, I will just mention one innovative feature of certified Kubernetes that I'm not sure folks are clear on. I should two. One that's particularly neat that works for this

approach is that in order to get certified, all you need to do is run this test suite. [Inaudible 0:43:29.7] is the test harness and then all these tests are already built into the repo and upload the results. One of the magical things about Kubernetes is that you could lie about that, right? You could just fake your logs, but then any of your users can come along at any point and rerun that test suite. If those results were different, they could report that.

Now obviously in most cases it would just be a bug, or some incompatibility and such. We have a whole process in place to deal with that and give them time for adjusting it and such. There is actually work in place to catch bad actors if that occurred. There's a distributed crowd sourcing aspect of how that works, that's very different from saying, “Oh, CNCF is the official testing lab and you have to send your software to us. Of course I'm old enough to remember mailing CD-ROMs in and other sorts of things, but this is a much more modern approach.

Another thing that we were genuinely worried about is someone saying a year ago, “Oh, well I'm certified to Kubernetes 1.7, so I'm certified Kubernetes and then just falling off the release train.” Of course, there's so much engineering effort that's going on to add new features to fix bugs, to improve the platform going forward. One of the other innovative aspects of the program is that if you continue to release your product at least once a year, so 1.7, 1.11, 1.15, then all of your old certifications remain compliant. In reality, it would be problematic after too long because the security bugs wouldn't be – security fixes wouldn't be available. In principle, you could back port security fixes and have much older releases on it.

In order to keep saying, “Oh, my older 1.7 release is certified Kubernetes,” essentially you have to have a 1.11 as well. If you don't, if you don't release one within a year, then that 1.7 certification goes away. Both of those are we think somewhat innovative features, but it's also trying to learn from the past.

[0:45:29.8] JM: To get back to your talk, again how good is our code, the idea that testing is really important, the ICD is really important, even for a project like Kubernetes, or especially for a project like Kubernetes. One reason for that is because of vulnerability in Kubernetes could be roughly as detrimental to internet infrastructure as a bug in Linux would be. If we got one of those vulnerabilities, do you have a framework in mind for incident response?

[0:46:00.4] DK: We do. We already did. There was one called subpath, about two months ago that was a very serious bug. The Kubernetes community has a Security Response Team. Really the CNCF's role is we just fund some private infrastructure for them to communicate with each other, but these are some of the very top security experts from some of the biggest companies and startups around the world. They get reports on it, and then they go investigate it and they find patches and then there's a release distribution mechanism. That piece, I mean, I certainly don't want to make it out as, "Oh, this is perfect and can't be improved or anything," but it's something – I mean, there's a lot of companies and a lot of money depending on Kubernetes. They take that work very seriously. CNCF's role there is just to facilitate it.

I will mention some other work that we've done that's a little newer, where two CNCF projects; Core DNS and Envoy requested that we fund us third-party security audits. We used a company Cure53 and they went off and Core DNS they found a serious bug that they fixed, but more generally they had some other suggestions and fixes, all those were taken into account, and the reports are public, so you can you can link to them in the show notes.

That's some work that we're interested in doing for more of our projects. I mean, security audits they're only ever a point in time, it's still humans doing it, so there's nothing perfect about it. We're really thrilled to be able to contribute some of the resources of our members back into those projects and to help them along. I will say that overall, the summary of the reports were extremely positive, that they spoke very highly in terms of the quality of the code and the conscientiousness of the design and did recommend them.

[0:47:45.0] JM: We started off talking about serverless. That's one of the working – I believe it's a working group, right? Yeah, it's a working group within the CNCF. Another working group that I attended a couple talks on was storage. Storage interesting because, well maybe you could just tell me, why you believe storage and storage on Kubernetes is interesting.

[0:48:06.3] DK: Sure. First of all, I'll just say storage run Kubernetes and more generally storage on distributed systems is hard.

[0:48:11.7] JM: Yes.

[0:48:13.0] DK: Everyone agrees that stateless apps of which serverless wants and tend to be, are easier to do, and stateful ones are harder. The storage working group has unfortunately been a little contentious, and there's been some innovative new startups and then there's a lot of traditional players that are saying, “Hey, we want to be able to hook our big existing hardware into clusters and make that available.

Thankfully, they have had some progress. There's a initiative, the container storage interface, which is essentially an API. It's exactly modeled on the container network interface, and the container runtime interface. That's now gotten adopted into Kubernetes. I think it's a beta level. That has been some success on it.

Then we do have our first project in CNCF. It's a sandbox-level project called Rook, which currently takes the Ceph distributed file system, which is this decade-old piece and then hooks it in a much more Kubernetes native way. I think this is one of those areas developer tools, where there's just a ton of innovation going on. I definitely don't expect there to be one solution for storage, or for distributed storage, but we're happy to facilitate the different players in the industry looking at different ways of working together.

[0:49:30.1] JM: Do you expect to see entirely new databases and storage styles come up once these primitives are built within Kubernetes, or do you think people will just start implementing things that cloud providers have?

[0:49:43.1] CA: Definitely we're expecting cloud native databases. Three that I'll mention of the test is a new incubating level project in CNCF, so relatively mature. It comes out of YouTube, where it was in production for seven years. It's a sharding technology from MySQL, which is being used by Slack, being used by a bunch of other really interesting companies and is very serious solution.

Two startups that aren't involved in CNCF directly but that we have good chats with our cockroach labs, which has essentially a PostgresSQL-compatible distributed database that's basis some degree on Google spanner, and one out of Beijing that I think is very interesting called TiDB, TiDB that's a MySQL compatible API and a complete distributed one.

I will give another quick shout-out to l.cncf.io. If you go to the database and data warehouse section there, it's a little crazy that I think we're up to 39 entries in there of different levels of maturity and scalability and distributed, but there's tons and tons of innovation. In fact that one last shout out I'll give is RethinkDB. CNCF helped get out of chapter 11 and get re-licensed to Apache 2.0. It's now Linux Foundation project, but I know a lot of folks have had a good experience with that. There are another 36 beyond the ones I mentioned.

[0:51:03.5] JM: All right. Well, Dan Kohn thanks for coming on Software Engineering Daily. It's been really great talking to you.

[0:51:07.2] DK: You too. look forward to seeing you in Seattle, December 11th to 13th.

[0:51:11.3] JM: Yes.

[0:51:11.6] DK: Hopefully maybe in Shanghai, November 14th and 15th.

[0:51:15.3] JM: I'm planning on both of those.

[0:51:16.1] DK: Awesome. Thank you.

[0:51:17.3] JM: Okay. Thanks, Dan.

[END OF INTERVIEW]

[0:51:20.7] JM: Your company needs to build a new app, but you don't have the spare engineering resources. There are some technical people in your company who have time to build apps, but they're not engineers. They don't know JavaScript or, iOS, or Android. That's where OutSystems comes in. OutSystems is a platform for building low-code apps.

As an enterprise grows, it needs more and more apps to support different types of customers and internal employee use cases. Do you need to build an app for inventory management? Does your bank need a simple mobile app for mobile banking transactions? Do you need an app for visualizing your customer data? OutSystems has everything you need to build, release

and update your apps without needing an expert engineer. If you are an engineer, you will be massively productive with OutSystems.

Find out how to get started with low-code apps today at outsystems.com/sedaily. There are videos showing how to use the OutSystems development platform and testimonials from enterprises like FICO, Mercedes Benz and Safeway.

I love to see new people exposed to software engineering. That's exactly what OutSystems does. OutSystems enables you to quickly build web and mobile applications, whether you are an engineer or not. Check out how to build low-code apps by going to outsystems.com/sedaily.

Thank you to OutSystems for being a new sponsor of Software Engineering Daily. You're building something that's really cool and very much needed in the world. Thank you OutSystems.

[END]