EPISODE 345

[INTRODUCTION]

[0:00:00.40] JM: Cloud computing changed how we develop applications for the web. Over the last decade engineers had been learning how to build software in this new paradigm. The costs have gone down but our nodes can still fail at any time. We no longer have to manage individual servers but the layers of containerization and virtualization require that we build new strategies for communicating between services.

As we've adjusted to this new way of building applications the term "cloud native" has become a useful descriptor. Cloud native applications are modern distributed systems that are capable of scaling to tens of thousands of self-healing multi-tenant nodes. Open-source projects such as Kubernetes, Prometheus and a linker-d draw on the lessons of the big technology companies that they were built at, like Google, or Twitter, or Soundcloud, which are the organizations that originated the strategies that turned into these open source projects.

Engineers today don't need to reinvent the key infrastructure of Google or Twitter. We can combine the open-source cloud native technologies and use them to build powerful systems more quickly. Dan Kohn is the executive director of the Cloud Native Computing Foundation and he joins me for an interview about the projects within the CNCF, how they fit together, and the future of computing, both enterprise and consuming.

[SPONSOR MESSAGE]

[0:01:42.40] JM: For more than 30 years, DNS has been one of the fundamental protocols of the internet. Yet, despite its accepted importance, it has never quite gotten the due that it deserves. Today's dynamic applications, hybrid clouds and volatile internet, demand that you rethink the strategic value and importance of your DNS choices.

Oracle Dyn provides DNS that is as dynamic and intelligent as your applications. Dyn DNS gets your users to the right cloud service, the right CDN, or the right datacenter using intelligent response to steer traffic based on business policies as well as real time internet conditions, like the security and the performance of the network path.

Dyn maps all internet pathways every 24 seconds via more than 500 million traceroutes. This is the equivalent of seven light years of distance, or 1.7 billion times around the circumference of the earth. With over 10 years of experience supporting the likes of Netflix, Twitter, Zappos, Etsy, and Salesforce, Dyn can scale to meet the demand of the largest web applications.

Get started with a free 30-day trial for your application by going to dyn.com/sedaily. After the free trial, Dyn's developer plans start at just $7 a month for world-class DNS. Rethink DNS, go to dyn.com/sedaily to learn more and get your free trial of Dyn DNS.

[INTERVIEW]

[0:03:39.9] JM: Dan Kohn is the executive director the Cloud Native Computing Foundation. Dan, welcome to Software Engineering Daily.

[0:03:46.0] DK: Thanks very much. Great to be here.

[0:03:48.0] JM: Now, the cloud computing phenomenon has really picked up and it started with the cloud providers such as AWS and since then it has evolved into a business that involved Docker, it involves Kubernetes and it's really changed the way that companies that are cloud native that grew up on the cloud are operating as well as the older companies who are moving to cloud. What are the most salient differences between the companies that are cloud native and the companies that did not grow up on the cloud?

[0:04:33.6] DK: I think that's a great thing to look at. If you look at most of our members of the Cloud Native Computing Foundation, the companies they're selling to and our end-user supporters who are examples of this, actually very few of them started out as truly cloud native. Even Google, that you could argue originated a lot of the ideas for containers and such, didn't actually start everything with containers. I think there's a real sense in which if your goal is just to pitch and say, "Hey, this is the best technology ever if you're starting from a complete greenfield." Then you're really trying to solve way too small and uninteresting of a problem. That really we think a huge value of Kubernetes and the other technologies within Cloud Native Computing Foundation is how you can migrate your legacy monolith, that sort of previous super large Java application, or Rails, or anything else and evolve it into a cloud native architecture.

Then to actually get to the answer, we define cloud native as dividing your application up into different pieces that are called micro-services wrapping each of those in its own container and then orchestrating those containers together. We really do think that it's both legacy technologies and nuance and new greenfield, so greenfield and brownfield can get a huge benefit from a cloud native approach.

[0:06:01.7] JM: Does cloud native refer to the fact that open source projects under the Cloud Native Computing Foundation, these projects themselves are cloud native?

[0:06:12.4] DK: No. I think it more refers to the idea that — There were all these technologies that were, as I said, monolith and have been running traditionally in these large data centers and then there's an approach of called lift and shift of how you can get it up into the cloud. There's often a ton of benefit from doing that in terms of redundancy and lowering costs and better bandwidth and other sorts of things, but the idea that that's really not enough. In a lot of ways that's just the very first step. Then making that application cloud native tends to involve trying to shave off pieces of it and split it into different parts that can be developed and managed and iterated on separately instead of having to just deal with the monolith.

Now, one of the huge advantages of cloud native is that it entails adopting open source software as the infrastructure and so that open-source software helps avoid lock-in. One of the goals here is that a lot of companies are locked into their data center today, they have a very high cost. They have a lack of flexibility. They have this goal of saying, "Hey if I could lift and shift and get it up into the cloud then I could get all these advantages," and that's true, but if they then lock themselves in to one specific cloud provider then they've actually given up a ton of those advantages. The biggest one is a year after they arrive when they're trying to renegotiate their contracts, they're not in a good position to be able to get those crisis down.

Where by contrast if they pick an open-source stack, such as one based on Kubernetes or other cloud native technologies, then they have the ability to migrate that cloud native application to any cloud. Also, they can even do that in the existing data center, Kubernetes and other cloud native technologies work great for private cloud, public cloud and also increasingly for hybrid cloud technologies where you, for example, burst your capacity in the public cloud even if the base is in your own private cloud.

[0:08:15.7] JM: Another aspect of the legacy migration to cloud native is from my interviews I've noticed that when a company does this legacy migration, to Kubernetes and Docker and the related stuff, there is a rejuvenation within the engineering team that can happen because sometimes you have these companies that are not only hobbled technologically, but when you get hobbled technologically it can break your spirit and it becomes harder to be optimistic about anything. When you have this roadmap that's fairly straightforward for migrating to a place where many of your operational problems will be solved, many of your innovation engine problems will be solved, it can rejuvenate the spirit of the company as well.

[0:09:12.0] DK: I completely agree, and I've actually lived this myself where I was involved in a startup company — It's a startup. It's not like a 50-year-old company or anything. It's not like they had [inaudible 0:09:23.0] or anything, but we just taken on a ton of technical debt in order to get the initial version of the product ad as a lot of folks do, probably even the majority. We had a monolith that was extremely challenging to evolve. We didn't have good test for it. All of your sort of clichés, the use of HPF that was very much a pet and not cattle.

A key part of the cloud native architecture is this insistence on continuous integration and continuous deployment that all of your infrastructure is programmatic, so this this concept of infrastructure as code. Once you make the investments that that — It's an investment. It's not trivial, but there are now fantastic tools available to help you along that path. Once you make investment, it is such a magical thing to see organizations that were previously stuck saying," Hey, I had this bug, or a customer reporter but. I created a fix for it. Maybe we can get that fix deployed in a month or in two months to a scenario where you can be doing dozens of deployments per day. Just, as you said, the sort of lifting spirits that that can bring.

[0:10:33.6] JM: You talked about the ability for a company to move their entire infrastructure off of a cloud provider to a different cloud provider if they get everything on to Kubernetes or even everything into just Docker containers because it gives you an architecture that will run predictably on different cloud providers. What do you think that's going to do to the cloud provider market as a whole?

[0:11:06.7] DK: In principle, it's going to increase competition, and that's a good thing for end-users, but I think it also just gives each of the cloud providers an incentive to find new avenues of innovation. For example, if you look at the way that Google is particular investing in graphical point unit, at its GPUs as an add-on or the way AWS has these pretty amazing services in and obviously to point LAMDA as a serverless technology to get people excited.

I think it's all good when you give customers more options. I think the huge magic of open-source is this concept of a people being able to invest together and share some of these core infrastructure questions without necessarily being locked in to the same vendor for years and decades to come.

[0:11:58.2] JM: I agree with that, and if there is not a mechanism for locking people into the core infrastructure, it dis-incentivizes these cloud providers from making these lock-in kinds of services and it leads to specialization which is more efficient. Microsoft can specialize in something. Google can specialize in something. Amazon can specialize in something, and it's not a zero-sum battle. It also, I think, encourages multi-cloud architectures.

What are the competitive advantages or the differentiating ways that you're seeing the different major cloud providers evolve their cloud offerings?

[0:12:44.7] DK: I think that it's very early days in this space. When I look at it, certainly AWS has invested a ton in creating a lot of really custom amazing services that have provided a huge amount of value, but when you adopt those, it does essentially prevent you from considering migrating to another cloud or it becomes much more challenging.

I think Azure obviously has a huge leg up in terms of their experience with Window, but I would suggest that Google and IBM, [inaudible 0:13:18.5] and AWS and others are investing there to make themselves competitive. When you look at Kubernetes, we are also making investments to be able to run Windows containers as well. I don't know that I have any brilliant insights into it. I think it's interesting to mention Alibaba cloud that they're a huge new player in the space, and so I'm not quite sure what the total number will be, but it is incredible to see all of these different avenues that folks are looking for innovation.

[0:13:50.5] JM: Let’s talk about some of the projects that are under CNCF. For people who are not sure what CNCF is, it’s similar to maybe the Linux Foundation or the Apache Software Foundation. It’s a collection of open-source projects and the CNCF provides some oversight and some economies of scale for putting these different projects in one place. Some sense of standardization. Some sense of governance, and we’ve covered that in some previous episodes around the CNCF. We’ve covered Kubernetes in detail on different episodes. We’ve covered Prometheus. We've covered open tracing a bit, or we covered Open Zipkin. People who are interested in those three projects can go check out those episodes, but since I've done the last shows on CNCF, there have been more projects have been added under CNCF and I'd like to go over those and then talk about how they fit into the overall direction that CNCF is going in.

One of them is Fluent D, which is an open-source logging solution to unify data collection and consumption. Why does open-source logging makes sense to come under the CNCF?

[0:15:04.2] DK: I think Fluent D is an amazing project. The way to think of it is just as you deploy your set of micro-services and your cloud, you can kind of take two approaches. You can take the Google approach where all the software that they write they require them to use a standard logging forma and that works great if you have that kind of thousands of developers and can kind of mandate rules like that.

For most of the rest of us, we’re dealing with a lot of different projects both open-source and close-source in our own internal proprietary ones and others that each of which have slightly different log formats. Also, that when you receive those formats, that you have different systems that you want to be able to log them and parse them and process them and such. You face the M times X problem where in principle, to adapt from any of the of 300 inputs to all the outputs, you just need a massive number of different adopters. Fluent D solves that but putting itself in the middle and being able to convert essentially from and to any format.

In a lot of ways it’s a very natural complement to open tracing which is a format and a standard for how tracing messaging can work that's been implemented, as you mentioned, by Zipkin and also by some other tracing frameworks, and also with Prometheus, which is a monitoring app, but tracing, monitoring, logging are all kinds of different versions of how do you check the health of your applications. How do you respond to changes on it. How do you know what's going on? All of those are basically central challenges for any kind of cloud native setup.

[SPONSOR MESSAGE]

[0:16:52.2] JM: An in-house team of engineers spent thousands of hours developing the Casper Mattress. As a software engineer, you know what that kind of development and dedication is like. The result is an exceptional product, one that you'd use and recommend to your friends. You deserve an exceptional night's rest so you can continue building great software.

Casper combines supportive memory foams for a sleep surface that's got just the right sink and just the right balance. Plus, its breathable design sleeps cool to help you regulate your temperature throughout the night. Stay cool people. Stay cool.

Buying a Casper mattress is completely risk-free. Casper offers free delivery and free returns with a 100 night home trial. If you don't love it, they'll pick it up and give you a full refund. Casper understands the importance of truly sleeping on a mattress before you commit, especially considering the going to be spending a third of your life it.

Amazon and Google reviews consistently rank Casper as a favorite mattress, so try out. Get a good night's rest and up-vote it yourself today. As a special offer to Software Engineering Daily, get $50 towards any mattress purchase by visiting casper.com/sedaily and using code “sedaily" at checkout. Check out casper.com/sedaily, reward yourself with a great mattress and a great night sleep and use promo code “sedaily” at checkout to get that $50 towards a mattress purchase. Thanks again to Casper.

[INTERVIEW CONTINUED]

[0:18:54.4] JM: Linker-d is now under the CNCF as well. This is a service mesh. It's a proxy for services. We did a show about linker-d a while ago with William Morgan. He's coming back on the show soon to talk more about linker-d. We also did a show with Matt Klein from Lyft about Envoy which another proxying application, and proxying is where you add this application on every node and every request that goes between two nodes passes through the service proxy layer. It adds this layer of standardization between every request and every communication between different services. What are the advantages of that standardization?

[0:19:48.3] DK: Sure. When I think of it, it's a very natural add-on to Kubernetes and other orchestration platforms like Mezos and Swarm. The way I think of it is in a completely vanilla installation where you're just getting things up and running and your sort of just playing with it. You just start with pure Kubernetes and you can use that to run your batch jobs. Then if you are sort of running some competition tasks of different requirements and then some batch jobs and some that have network I/O and Kubernetes is very smart about allocating out all the containers and all the pods in such a way as to try and maximize your resources, and all that works great.

Then just as things get a little bit more complex and say you're offering a web service to access some of these things and now you start worrying about, "Oh, wait a minute. What happens if not just my whole pod dies but if certain of the request or servers get overloaded? Now, how can I redirect on English, for example, to automatically send the request to some of the more likely mode of servers?" That's just sort of the most obvious example from there where you have a micro-services architecture and there's multiple different parts, you can imagine different kinds of congestion occurring or different kinds of feedback that you want to get.

Really, the magic of linker-s, and as you mentioned, Envoy and these other projects, is this ability to add metrics in to your — The hops along the way that you can then intelligently decide, "How am I going to respond to those metrics Oh, I see the latencies going up on the server. I'm going to redirect the request to these other servers until the latency goes down again. It's such an amazing level of customization flexibility around all of those sort of more complex use case.

[0:21:42.3] JM: Kubernetes is essentially the cluster manager for these different containers that you have running micro-services. What's the relationship between linker-d and Kubernetes, because you are talking about load-balancing here, I would think of Kubernetes as the manager for how load is being distributed, but maybe I'm wrong about that. Help me understand the connection between the service proxying layer and the orchestration layer.

[0:22:12.8] DK: Sure. Linker-d is very much designed to work with Kubernetes. It's not a substitute network. The easiest way to think of it is providing a ton of additional information back to Kubernetes and to the English controller which is the sort of central point where the traffic comes in from the internet to allow those orchestrators to decide how to allocate things out. It is very much designed as a complement, it's not an alternative, or its additional features on top.

[0:22:44.1] JM: GRPC is another component of the CNCF stack. We've done a couple of shows about GRPC, but explain why GRPC is so useful to the cloud native architecture.

[0:22:58.4] DK: Sure. We're now up to nine projects in CNCF and we kind of got GRPC in there the same month that we brought in container stuff and other sorts of things and I worry sometimes the people don't notice it because it is very much plumbing. It's not a real high-level sexy project that users interact with the way, say, Prometheus or Kubernetes is. I think it's really one of the most fundamental infrastructure and connective tissue projects that we have and we're really just thrilled that the GRPC team chose to get hosted in CNCF.

The simplest possible way to think of it is a ton of web apps out there use JSON plus REST. They use JSON as their format for sending information and they use REST for the standard web commands of how you deal with that. That works completed fine. It absolutely meets people's needs until it doesn't. At a certain level of performance and traffic you want to move away from those kind of text-based protocols that involve a lot of parsing and necessarily are just slower to much more efficient binary protocol.

Within Google, GRPC comes out of Google, it builds on an internal library that they used extensively called Stubby, and GRPC is essentially the next version of that where they've done just a ton of thinking and incredible real-world use cases around it. In particular, they were able to make the protocol smaller because it builds on top HTTP 2. HTTP 2 just has a ton of features in it in terms of how it handles streaming, binary directionality and other kinds of things too, and then GRPC uses protocol buffs. These protocol buffers as the binary format.

The idea then is that you define the messages pretty similarly to the way that you would define your JSON payload and then GRPC can build those out and there're libraries available for basically every programming language, and then there's also things like gateways if you still want to have a JSON REST interface for users who are uncomfortable yet with GRPC.

What's pretty neat about GRPC is that it's potentially the native communications mechanism being used by Kubernetes. It's also used extensively within Docker Swarm, within Container D, another one of our projects. There's just a ton of overlapping commonalities between that and the other projects. We're also just seeing a lot of very big reputable companies and also startups adopting it internally just because they see that the libraries have ton of investment in them, large communities they've been well debugged, that it really is just a very positive approach for how you do a remote procedure call.

[0:25:58.2] JM: Here's a question that I'm wondering about GRPC. When I make a request to some API on the internet, like the yelp API. If I want to make a request to the yelp API for a list of restaurants in certain geo-domain, I'm typically sending a request that is JSON over REST and — I don't know if that's the right way putting it.

[0:26:23.0] DK: No, it is.

[0:26:24.0] JM: Okay. JSON over REST. Why am I not using GRPC for that? Why am I not using GRPC for all of my communications throughout the internet?

[0:26:33.6] DK: The answer is just that JSON REST has around for 10 years longer. If we had this conversation nine years ago, you wouldn't be doing JSON REST, you would be doing XML over REST. Before that we did get back to the bad, scary WS Star days, and before that CORBA.

It's not a non-invented hear syndrome, it's a non-invented yet syndrome where my expectation overtime is that you're going to see more and more companies like Yelp and — I don't think Twilio, or other — Other sort of developers, centric services offer a GRPC endpoint. I think it's still probably a while before they removed JSON REST just because of it so ubiquitous and well-known.

The other thing is everybody would love to be doing thousands of you Stripe transactions per second, but most of our businesses probably aren't interacting with them like that volume. Starting out with JSON —

[0:27:36.6] JM: Each message is not that big. You're talking about GRPC having the main advantage of you can serialize and de-serialize these big, heavy, complex messages quite quickly, most of the messages that you're probably sending to Stripe are fairly standardized.

[0:27:53.5] DK: Again, we all hope to create the startup where it's hundreds of thousands of requests to Stripe per second at peak times when you turn on slash sale. How many of us actually have to deal with that is maybe an open question.

What I will say as an example is an early version of Kubernetes was using JSON for a lot of its messaging, there actually were congestion in other issues, and so the evolution to GRPC was a pretty natural thing. Obviously, you can make it work with JSON. You could always just move to beefier machines and beefier networks and such, but the whole point is you don't want to, is that you'd like your machine to be up to focus on the actual task it's supposed to be doing and not just communicating its status with the rest bolster.

[0:28:41.4] JM: Yeah. Core DNS is another project under CNCF, and this provides domain name service discovery. What is different about DNS in a cloud native world?

[0:28:53.7] DK: Yeah, it is DNS, and what's a little funny about core DNS is it's so incredibly flexible that there's sort of no limit of what you can do with it. I would definitely emphasize the service discovery aspect where the idea is that for a ton of real-world use cases, you're having your different microservices each being given a name within your cluster.

I've heard this statistic out of Uber that — Just a couple of years ago, but that they had 1,300 engineers, but 4,000 different micro services. You can imagine overtime you don't — You want to come up with meaningful names for each of those services and then you want those names to dynamically reach each of the services. Interestingly some of the things that we're talking about with linker-D, you can actually do with DNS in terms of having things expire and how you do Round Robin selection of your next server to talk to in such. Linker-D gives you a ton more flexibility.

Core DNS, it's based on a very modern architecture, so it's interestingly — It's actually built on top of the Go webserver which has a ton of users and is widely debugged and very performant. Then instead of doing HTTP or HPS, it provides these DNS records. Then it just has — Because of the caddy architecture, an insane amount of flexibility as to what you use as your backend so you can have SED, or you can have files, or you can proxy to an upstream recursive server, and then it has a ton of middleware built in that you can produce Prometheus metrics and do different kinds of load-balancing. Of course, you can always write your own.

I think we're vaguely hopeful that the Kubernetes team will at some point decide to adopt Core DNS instead of the system they ship now, Cube DNS. Whether they do or not, it remains a super useful project for more complex installations that are looking to use leverage DNS for more complex projects and service discovery.

[0:31:07.3] JM: On the container's side of things, there is Container D which is an industry-standard container runtime, and rocket, which is an application container engine. Explain the roles of the container runtime and the container engine in a cloud native architecture.

[0:31:28.8] DK: Yeah, fundamentally, they're really doing the same thing, which is that Kubernetes was originally designed around Docker containers. Interestingly, it actually was originally designed internally with Google around a more generic kind of container, but then when they saw the incredible adoption uptake of Docker, they focused it on being able to run Docker containers really well.

More recently, there's been a lot of work within Kubernetes to create what's called the container runtime interface, CRI, and essentially that allows you to use different kinds of containers of which Container D and Rocket are by far the two most common, the ones that you'll most likely go to use. You could say, "Why wouldn't I just use Docker?"

The answer is that Docker includes a lot of extra functionality that you need on your desktop and probably in your continuous deployment server as you're building these containers, but you don't need just to run the container. You need the ability to read the image, but there's definitely no requirement to write an image.

Docker, when it was originally developed, had a much more monolithic approach. What the Docker community has been focused on over the last six months, several months, is to split up Docker into a set of services of which Container D is that core runtime upstream that is then used within Docker. Rocket is an alternative that. It originally comes out of Core OS and has a lot of very valuable engineering from Core OS and Kimble in particular. The two different approaches, different code bases to accomplish the same thing which is a software for it to actually load and run the container within a Kubernetes pot.

[0:33:21.6] JM: All of these different projects fit together a way that makes sense. It's quite applicable to many architectures out there and it starts to look to me almost like how in the Linux you have these specific applications that do really basic utilities, like sed, or awk, or htop, and I'm wondering if you're starting to see a vision for some sort of platform that brings all of these things together. I originally thought that that platform was Kubernetes. What I'm starting to see is Kubernetes is just one part of all of these different projects that fit together.

What are you seeing in terms of the layer that is composing all of these things together, or is that Kubernetes?

[0:34:17.0] DK: No, I think it's a great question, and I will say — The Cloud Native Computing Foundation is part of the Linux foundation and I was the chief operating officer at the Linux Foundation for five years from 2006 to 2010. I certainly think about and look at these analogies all the time, and our Executive Director; Jim Zemlin just recently gave a couple of speeches saying, "Hey, Kubernetes is the Linux of the clouds."

That, as a sort of idea or a message point, is not something that I would argue against or have any concerns about. I probably would draw the analogy to Kubernetes being something like the Linux kernel, and then to say that most users out there are going to want a RHEL, or a CentOS, or a SUSE or Ubuntu that combines together multiple different projects and ensures compatibility and provides support for them

I think one way of thinking of CNCF is that we are trying to identify and help along several of the key projects, and so I don't think any of these analogies are perfect, but if you could think of Prometheus as, as you said, the h top or one of these being the [inaudible 0:35:31.2] and that the project do work well together. Just as in upstream or in the Linux world, different environments call for different combinations of projects. Obviously, android phones that are based on Linux use a different set of libraries and different set of packages than your typical cloud server or a desktop Linux for that matter.

We're definitely not trying to dictate in any way that you must use these sets of projects together. One way we've described what we do — We have this document called the Cloud Native Landscape, and you can see you that github.com/cncf/landscape, and it's a little overwhelming because it's just 448 icons right now representing all the different projects and also close-source offerings from different companies. Then below it we sort of gray out most of those and show the projects that are currently — The nine projects that are currently in CNCF and some of the other ones that we're looking at.

One of the metaphors we've used to say that with that landscape and other work we're doing, we're trying to help build the map of the cloud native landscape and then we're trying to share one or a couple particularly good paths through that terrain, but we're not trying to tell people, "This is the only way to get there."

I think in a lot of deployments out there they'll use several CNCF technologies and then maybe they'll swap in their own internal product for one of our projects, or maybe they'll pick a promotional offering instead of one of our projects, and that's a totally reasonable way. Our message is that there's multiple different paths to get to the same destination.

[SPONSOR MESSAGE]

[0:37:17.2] JM: Artificial intelligence is dramatically evolving the way that our world works, and to make AI easier and faster, we need new kinds of hardware and software, which is why Intel acquired Nervana Systems and its platform for deep learning.

Intel Nervana is hiring engineers to help develop a full stack for AI from chip design to software frameworks. Go to softwareengineeringdaily.com/intel to apply for an opening on the team. To learn more about the company, check out the interviews that I've conducted with its engineers. Those are also available at softwareengineeringdaily.com/intel. Come build the future with Intel Nervana. Go to softwareengineeringdaily.com/intel to apply now.

[INTERVIEW CONTINUED]

[0:38:14.2] JM: We've been talking about infrastructure that enterprises would use. We started this conversation discussing moving an enterprise monolith to the cloud, breaking it up into services that can scale on demand. It's in containers. Sometimes when I'm thinking about this space, I think about my usage as a consumer. When I use my Apple laptop and when I run out of space on my hard drive or I ran out of memory and my computer starts to perform poorly, that's a really frustrating experience.

It feels very antiquated experience and it would be so much nicer if somehow my UI was just a window into a Kubernetes cluster and whenever I opened up an application, if it was pressing up against the upper bound of my available memory, then my Kubernetes cluster would just scale up and I would get to continue my life as normal. When are we going to start to see these technologies make their way into consumer products?

[0:39:23.6] DK: Yes, I am going to swear off of promising that next year is the year of the Linux desktop, because I have been involved in Linux for a bunch of years and I still don't think we're there.

[0:39:35.3] JM: It doesn't even have to be Linux, right?

[0:39:36.9] DK: No. No. I understand. Let me make this argument to you, which is that you and I are getting somewhat rare these days and that we actually use laptop and desktop computers in our power users in that sense. The vast majority of people out there are using their smartphone as the primary way to interact with the information technology. For those people. it actually is incredibly rare for a modern android or iPhone to lock up or to you start chugging along.

Yes, maybe need to power cycle it every week or two in the worst case, but even that I think it's getting pretty rare. I'd say a lot of those people already are making use of those services that you're talking about, that all the key data they care about hope — If it configured things right, hopefully they have, lives in the cloud. If their phone dies, hopefully that they can just replace it under warranty and get a new one and maybe it takes a couple hours for it to sync up, but eventually it gets back to exactly the state that they had left it in.

If they're using Pokémon go or eBay or Ticketmaster or buying things from Nordstrom, to call out to four of our sort of well-known users out there. In all of those cases they are using Kubernetes and Prometheus and several of these other technologies that we're talking about. They just obviously don't know it.

[0:41:02.4] JM: I saw a video from Jeff Barr who is the chief evangelist at Amazon Web Services. He posted this video where his workflow was he brought his smart phone to his desk and then there was a dock on his desk and he does put a smartphone in the dock and the dock was connected to his monitor and what happened was his smartphone opened a connection to a VM on AWS and basically his smartphone was serving as his desktop computer and his monitor was just connectedness to his smart phone which was connected to a VM. He was using a smartphone is that as a portal into a VM. I thought was like a pretty cool model. I guess where we're going to go is eventually we just have like VR or AR goggles or we have some kind of hologram display or something but it seems like the monitors and that desktop computers are perhaps going away and we'll just have some easy way to do walk around with a smartphone and have it be the desktop computers. Is that sound like a plausible direction that the consumer could go?

[0:42:15.7] DK: Yeah. I'm a little skeptical if you need to plug in your smartphone. From my perspective, everything should be in the cloud and you just fix it down a new terminal and that environment. In fact, at my last startup, I replaced all of our customer service reps' Windows PCs which we're constantly having issues with; viruses and crashing and other sorts of things. With Chrome Boxes, which are just super cheap Chrome Box with the HDMI port for hooking to a monitor.

Literally, this morning, my fifth grader is working on a project on the Oregon Trail and he was upset that he couldn't use the laptop he's been using because it was someone else borrowing it. I said, "No. Just popup this Chrome Book and all the work you've been doing on Google Docs is exactly there," and he had that sort of Nervana experience of the exact last type was — It was right there as he'd left it. He was able to re-create the environment as. I think in a lot of ways it's more sort of our generation of folks who are stuck in the mud and have our laptops configured exactly the way that we want them that are the least flexible.

[0:43:24.] JM: It seems like the Cloud Native Computing Foundation Docker are turning into these two environments for doing cloud native type of stuff that are somewhat disconnected. There are elements of overlap between them, but it seems like — I haven't been covering this space as much lately as I was six months ago, but it seems like Docker has kind of like spun up some different projects on their own and they are may be not as closely tied to the rest of the ecosystem as they used to.

Could you give me a picture for how Docker as a company and Docker as a product is evolving relative to what you see in the CNCF?

[0:44:05.9] DK: Sure. Although I would argue that the directions actually gone very much the other way where I think Docker as a company is much more closely aligned with CNCF and a lot of our other members than they were, say, even a year ago.

[0:44:22.5] JM: It's good to hear.

[0:44:23.5] DK: Yeah, it is. That the context here, by the way, is that when Cloud Native Computing Foundation was spun up about 18 months ago, the idea was that Kubernetes was to be one the first projects to come in, and it was, but the goal was to have the foundation represent more than just Kubernetes. The belief was that it could add the most value by providing some standardization and common hosting and services across the cloud native industry. They were very intent and we're successful in getting not just major Kubernetes vendors, like Google and Huawei and IBM and Red Hat and Core OS as platinum members, but also getting Docker in as a platinum member and Mesosphere as a platinum member.

Docker obviously — Their orchestration platform is Docker Swarm and Mesosphere provide support for Mezos, which is the third config, and it was the first one in terms of first created for orchestration platform and originally came out of Twitter.

When you look at Docker year ago, they had a message about combining things together and bundling Swarm kits, their orchestration, into the base Docker image that a lot of folks had concerns about bec when you run software in a Kubernetes cluster or Mezos cluster elsewhere, you want to run the least amount of software, have the small service area if possible.

Just as of a few weeks ago, I guess, with ther announcements at Docker Con, they've announced this move away from monolithic where they've essentially re-factored Docker into a large number of different upstream projects, and Container D was the first step along that way, but that they now also have a Linux kit and Notary and several others.

Essentially that's telling different — Allowing different members of the cloud native community to make use of the pieces of Docker that are helpful for them. For tons of just regular companies out there, they can continue using the Docker CE, their consumer edition, which is essentially the free version of Docker that most people use. If you want commercial support for it, they have the Docker enterprise edition, but then you have more specialized cases, like people who just Container D to run at Kubernetes or now they've support these tools the Moby Rubric to allow you to plug these different components together in a way that's helpful for you.

I certainly would admit that there remains tension between Docker and some of our other members and I think that that's sort of always going to be the case for a foundation like ours that brings together different competitors and tries to find some areas where they can agree and then other areas where they compete very aggressively. As of now, we're pretty happy with the alignment that we've been able to achieve.

[0:47:22.6] JM: That's great to hear. The last I was covering this, there was that acrimony around people who are concerned that Docker was bundling in some stuff was, to a lot of people, who wanted to just run on Kubernetes, was essentially bloatware. If they've overcome that, then that's just fantastic because it's such a big ecosystem with so many business opportunities. There's really no need for people to step on each other's toes as aggressively as

— Maybe I misunderstood the narrative and I didn’t interview anybody from Docker around that time, so perhaps I didn't get the full story, but it just seemed to me I was — It seems like looking at a positive-sum ecosystem with a zero-sum mentality.

[0:48:06.2] DK: This is actually something that I spend a lot of time with and I’d say one of the fundamental strategic concepts of why CNCF was created is because we believe that the amount of investment in the cloud native space can be positive-sum.

When a big Wall Street bank decides on the vendor who's they’re to pay, support contract to, that ultimately is zero-sum and you can certainly imagine that many of our members are competing for that contract and very much want to win it and they can have some sharp elbows when they do it. If you look at a project like GRPC that is not just a core component of a core fundamental technology used in Kubernetes, but is also a core upstream technology being used in Docker. That investment in GRPC having a neutral [inaudible 0:48:58.2] for it, we helped them switch from a slightly unusual license they had to of our standard Apache license that just makes it easier for different people to contribute and to adopt it. All of those things we really see as being a positive-sum interaction.

[0:49:14.2] JM: Okay Dan, last question. From your time at the Linux Foundation, what did you take away? What lessons did you take away from building a thriving ecosystem that you are now applying at the CNCF?

[0:49:31.9] DK: I think the fundamental lesson that I took away was a round of maniacal focus on developers, if I sound like Steve Ballmer in that crazy speech he gave a decade ago. When you look at how the Linux Foundation was structured and is structured, nothing has actually changed about this, it has always separated the technology decisions from the business ones and it has never tried to dictate to developers how to do things or how to organize. We see ourselves as providing these kind of services to those developers.

Sometimes, like with Prometheus right now, the project is asking CNCF for help with governance and we’re working with them on creating a document for it. If you look at Kubernetes, they have their own internal process that they’ve created of a bootstrap governance committee that that's working through that right. While we’ve offered support to them and we've — For example, connected to them to Greg Kroah-Hartman to talk about how Linux is governed and some of the development processes we connected them with some folks from Node.js, which is another Linux Foundation project that's extremely high velocity, extremely successful to look at some of their processes.

We've offered help and facilitation, but ultimately what you see is that there's not a Linux Foundation way or a CNCF way that we try and dictates to them. We really do see yourself as — Or them as our customers and ourselves as service provider. That close over as well into conferences which is an area that we think we had a lot of value. In a fundamental way, we see the core developers of our projects as being the phytoplankton at the bottom of the food chain and when we get those people being willing to attend and participate and present at our conferences, then all of these developers from other companies that want to work with them, that want to contribute to them, want to be able to have hallway conversations come as well. Then when they're there, there's all sorts of vendors who want to provide services and get their message across to all of those other developers.

I was accused at one point of saying that phytoplankton might not be the friendliest analogy. I do actually mean it in a positive way and rather than saying that the developers are being devoured. Maybe we could say it's their knowledge that's being consumed.

[0:51:59.4] JM: Maybe the sand in the silicon, or the sand in the concrete. More of a foundational building block than a bottom of the food chain consumptive —

[0:52:12.7] DK: My wife is a oceanographer, so that may be biasing my choice in metaphors.

[0:52:18.4] JM: All right. Even the oceanographer, you could say like the molecular — It's just more — I don't know. Less victimizing. Anyway, Dan, thank you for coming on Software Engineering Daily. It's been a lot of fun.

[0:52:32.4] DK: Yeah, I really enjoyed the chat, and I look forward to seeing you at future events. I do want to just give a shout out for — It's still six months away, but our biggest even ever will be in Austin, December 6th through 8th is CloudNativeCon and KubeCon, and we definitely are excited about it and we'd love see a lot of your listeners there.

[0:52:50.8] JM: Yeah, I would too. I might go to that because I'm from Austin, originally. So I hope to be there.

[0:52:55.5] DK: Great. I look forward to t, and maybe we can do another podcast there.

[END OF INTERVIEW]

[0:53:05.4] JM: Indeed Prime is a sponsor of Software Engineering Daily and they worked with us to deliver some tips for hiring, for the hiring process. I wrote these tips myself but they are sponsored by Indeed Prime. Today's set of tips is about staying organized and motivated during a job search. We've all been there. We've all been searching for a job before. It's easy to get demotivated. It's easy to get lost and disorganized and I've got some tips about this process because I've gone through a number of times myself. Tip number one is do programming problems constantly. Whether you are at the job that you're about to leave or you are at home after your job, or if you're totally unemployed, just work on these programming problems.

The way that most job interview processes work is there is some point in it that you have to do white-boarding problems or coding problems online, and it's a stupid process, but that's the way that things work. You can actually have fun doing it. I actually kind of had fun doing it. I just think that they're really — It's insane and stupid way to hire. These becomes, like reverse a string, or find all the subset of all the numbers in a set of numbers that sums to 15." They're actually kind of interesting problems. They can be fun. They can be little game. You can time yourself. You can find better algorithms to do it. Whatever your mechanism for making it tolerable is, do them constantly.

Actually, these programming problems can make you a better programmer, and some people really like them. If you haven't taken them seriously, then there are plenty of articles about doing programming problems, competitive programming, cracking the coding interview and so on.

Number two, during a very job search, you should have a side project that excites you, that you schedule some time to work on, because it's really important to have something that connects you to the creative side and fun side of programming. So having a side project that you're going to work on maybe over the weekend or maybe every night during the week or two or three nights a week, you should schedule time to work on that side projects so that you stay fresh and interested in the general world of programming.

Tip number three is to listen to podcasts about software engineering. If you're at the end of the Software Engineering Daily episode, I probably don't need to be suggesting this, but there a lot of podcast that software engineering and some of them are about the software skills. Those can be helpful for staying organized and motivated.

Number four; in order to stay motivated when you're searching for a job and you're getting rejection after rejection, I find it can be helpful to listen to self-help audiobooks or to read self-help books. There's plenty of self-help books out there. I don't need to list any, but these things are great, *Seven Habits of Highly Effective People.* I'm listing them anyway, or *How to Win Friends and Influence People.* These things can be really really helpful in getting your positivity going, which is super important to project during job interviews.

Tip number five is to find a way to say what you mean and mean what you say. Don't try to deceive the companies that you're interviewing with, because it's a terrible way to set yourself up for a bad job and it's just not necessary. If you're applying for — You can find engineering jobs that are interesting enough to you if you're willing to work at sound engineering job.

Don't tell a company that you're not actually interested in. Don't deceive them. If you're not interested in the work itself, or you're not interested in the company that you're working for, that you're applying for, find a way to say something good about the specific project that you're applying. Maybe you're not interested in working for an oil company, but like the idea of working with big data and Internet of things data, an oil company has plenty of that.

Tip number six, buy white boards for your home or your apartment. I like to put up whiteboards all throughout my home because you can always practice programming problems on it. You can write self-help little quotes and notes to yourself and ways to stay motivated. I think whiteboards are great utility for somebody who is having trouble staying organized or motivated.

Number seven; research and deeply understand the companies that you are applying to. You should be able to list off 5 to 10 ways that you can improve the business that you're going to work at. This might sound strange to somebody who is just applying to be an engineer, “I’m just an engineer. I just take orders. Why does it matter what I understand about the business?” It’s actually a tremendously important that you understand the business and try to think of business value proactively rather than just being a cog. No company actually wants to hire a cog, although the hiring processes often suggest that.

Tip number eight is the final tip. Every day, think of the business that you can start or a valuable product that you could build. I think this is important for somebody that is having trouble during the job search process because thinking through alternative paths can make you a little more optimistic because sometimes you’re just not cut out for working for somebody else and you should build your own thing, you should start your own project, you should be entrepreneurial. You should think maybe that's you, but maybe it's not you today. Maybe it's you in five years or 10 years or 20 years, but I think there's no time like the present to start thinking of businesses you could build our products or open-source projects or just stuff you can manage yourself. I think this will give you an optimistic path that is independent of who picks you to work for them.

Those are my eight tips to staying organized and motivated during a job search. These tips are brought to you by Indeed Prime. You can go to indeedprime.com/sedaily to check out a new way to do a job search, to look for a job. Indeed Prime connects you to jobs that you would find interesting and that you would be a good fit for. Indeed Prime is much better than just searching for a job. It’s a new way to get hired or a new way to hire if you’re a company. Go to indeed.com/sedaily, and thanks to Indeed for sponsoring these tips.

[END]