Deloitte.

For Cloud Professionals
March 2019

For Cloud Professionals, part of the On Cloud Podcast

David Linthicum, Managing Director, Chief Cloud Strategy Officer, Deloitte Consulting LLP

Title:	**Containers and Kubernetes: standardizing cloud migration**
Description:	In this episode, David Linthicum and guests Justin Domingus of CareZone and Bitfield Consulting's John Arundel discuss DevOps, cloud migration, containers, and Kubernetes. David, John, and Justin tackle how organizations can use Kubernetes as a platform to power improved DevOps and make cloud migration more process-oriented and straightforward, while still operating in their unique environment. They also take a look at automated testing in containers and DevOps and the challenges associated with moving legacy apps into containers, and some myths and truths about containers, Kubernetes, and ROI.
Duration:	**00:26:17**

Operator: The views, thoughts, and opinions expressed by speakers or guests on this podcast belong solely to them and do not necessarily reflect those of the hosts, the moderators, or Deloitte. Welcome to On Cloud, the podcast for cloud professionals, where we break down the state of cloud computing today and how you can unleash the power of cloud for your enterprise. Now here is your host David Linthicum.

David Linthicum:
Hey, guys welcome back to the podcast. We've got some special information from you with our special guests who just wrote a book. Their names are John Arundel. I always – I thought I was screwing that up before we did the recording. And Justin Domingus. And, actually, Justin's the DevOps engineer at CareZone.com. Love to hear more about that. And John is a cloud native consultant for hire. I love that, a hired gun, a mercenary. Author, “Cloud Native DevOps with Kubernetes,” which is what we're going to talk about today. So, John, you first. Give us kind of the John Arundel story. How did you guys meet and work on the book and what you've been focusing on for the last five years. I love to get the bios of our guests.

John Arundel:

Sure, so yeah, I've been doing consulting work for about fifteen years now, and it's mainly helping companies get the most out of their infrastructure. And every client that I talk to has a lot of pain points with what they have. I try and fix those. And they also have questions about the future. What is happening with cloud and cloud native? Should I go there? How do I get there? That kind of thing. So that's what I do.

David Linthicum:
Justin, what about you?

Justin Domingus:
Yeah, so I'm a DevOps engineer in Seattle and been living here for about four or five years with my wife, who is a software developer. And I've mostly focused on the DevOps space, in particular with containers, build pipelines, things like that. I've worked at a few different startups, so usually focused on kind of small-scale organizations.

David Linthicum:
So, what was the brainchild of the book? Was this the way I write books, is you kind of start out with one concept and a title and end up with a completely different concept and a title, or this was purpose built for cloud native DevOps?

John Arundel:
Yeah, I think we probably went through three or four different concepts along the way. I'm sure you find the same thing. So, we really wrote this book because we needed it. You know, when we were looking at how do we move CareZone to Kubernetes, I was thinking about the same thing for other clients. You know, we have questions like, firstly, what's it all about and what difference is it going to make to the way we do DevOps and how do we get onto Kubernetes and where do we buy it from? Big questions like these. And once we've got Kubernetes, what in the world do we do with it? Assuming you have a Kubernetes cluster, what – how big should it be, and how many should there be, and how should you deploy your app to it, and should you have several clusters of – you know, there's a million questions. And we didn't know the answers to any of them, and we thought it would be great if there was a book which tells you, and there was not. So, we tried to fill that gap.

David Linthicum:
Yeah, it's funny. I wrote an article a couple years ago talking about DevOps and Kubernetes and containers, and it broke records, which tells me that the topic is hot, and everybody really is trying to figure out guidance. I'm not sure my guidance was the ultimate guidance you can get with a complete book, you know, with a 2,000-word article, but I kind of found that there is some modifications that need to be made to the DevOps tool chain in the processes and how you deal with people, and DevOps is all about people and processes, as I'm finding. So how did you guys approach it? What's changed? What's the delta between DevOps without containers and Kubernetes and DevOps with containers and Kubernetes?

Justin Domingus:
From my perspective, I think Kubernetes really just gives you a standard platform to talk around, like it's kind of – it answers a lot of questions that I think without it you have to kind of solve for yourself. So, if you're not using Kubernetes in a DevOps organization, you're going to have things like config management, you're going to use either Puppet or Ansible or something like that. You're going to have your own deploy scripts. You're going to have scripts that manage rolling deployments and how to (inaudible) logs and monitoring, so kind of all the little bits and pieces that you need for your software, you have to, as an organization, either build that or kind of bolt things together. I think Kubernetes just has kind of come up with a standard set of tools that's already bolted that stuff together and it's automated most of what most people want to do.

David Linthicum:
So, Kubernetes is itself a DevOps tool, or Kubernetes of itself is just an aspect of DevOps, enabling technology, so to speak? Build and deploy.

Justin Domingus:
I mean, a common phrase people use is they call it like the new operating system of the cloud and, you know, you can think about it as the platform you run your software on. Everyone used to say, well, we run our software on Linux or on Windows or whatever, and now people say, well, I run my software in containers and those containers run on Kubernetes, so they think of it as the operating system. So, I would say it's just a tool, but it can enable highly effective DevOps practices.

David Linthicum:
So, you guys got gung-ho in Kubernetes, and this is a question for you, John, that it is going to be the new platform of the cloud and we need to start migrating applications over there. And what about legacy applications written twenty years ago, ten years ago? Do those ever exist on Kubernetes, in containers, or is this something where we're just kind of layering in as kind of the next new shiny object where we're going to run our existing stuff?

John Arundel:
Yeah, absolutely. You just stick a new layer over the top, and there'll be a mainframe down there somewhere way at the bottom with a green screen terminal. But yeah, you're absolutely right. It's a deep question, and I think before you think about Kubernetes, you have to think about containers, so what does the container mean. And the thing is, it's a lot easier to answer that if you were around before containers, which some of us were, and you have this terrible problem trying to deploy your software places. You need machines to run it on and you need to provision those machines, and manage them, and manage the configuration, and you need to keep configurations synchronized across lots of machines and networks. And the bigger scale you're operating at, the more machines you need, and so the bigger configuration problem you have. And if you deploy your software to these machines, you need your stack, and you need dependencies, and you need databases, and language interpreters, and web servers, and all of these things. So that's a huge problem, probably bigger than writing software in the first place.

David Linthicum:
Now are you saying – sorry to interrupt you there – are you saying that those are going to exist within Kubernetes or outside of Kubernetes?

John Arundel:
Well, here's the thing. All of those dependencies exist. You can't get rid of those. The question is can you – how do you package them up with your software to deploy it. So, containers are absolutely brilliant for this, as you know, because you can put everything inside the container that the app needs to run. Then the problem just becomes one of, "How do I run a load of containers." And once you have things in containers, then you can say things like, "Could I make this automatically reliable by running lots of replicas of this service and failing over between them automatically?" Can I load share and load balance? Can I get metrics about what's going on with these containers?" The point is it doesn't matter what's in the container. You can put whatever you like, and that could be a legacy app, a Java app, a (inaudible) app, whatever you like, or it could be a native binary. The point is, from the operations point of view, it's just a container. You don't need to care what's inside it, so therefore you can use a standard platform. And for a long time, there wasn't one, so we kind of had a problem where I've got my containers, so now I still have to run them on my machines, and I still have to manage them. What I want is just a big old cloud of compute and I can just stuff my container in it and have it work. And we kind of have that now with Kubernetes. It's not that there's anything so magical about Kubernetes itself so much as the fact that it's just universally adopted now.

David Linthicum:
So, Justin, question for you. So, I'm looking at this in a bit more pragmatic way. So, I'm a global 2000 enterprise, and those are typically going to be our clients, and we're looking, in essence, to leverage Kubernetes within an existing DevOps chain, or moving into the cloud. We have an existing 3,000 applications, thousands of them are existing legacy things; they're going to be very difficult to move in the cloud and certainly not make cloud native. Two thousand are LAMP stack stuff, things built in the last twenty years, which are probably much more portable into the cloud. We're looking to move into the cloud, we're looking to move into DevOps, we're typically looking to move into multi-cloud, so the deployment targets are going to be moving and changing over time, and we are interested in moving to DevOps. So, give me the steps, basically the elevator ride pitch, and say it's a long elevator ride, 30 floors, in terms of how we're going to make the migration and transition in this kind of environment because a lot of organizations, kind of the as-is state of IT, it seems like mission impossible, but I'd love to get your guidance on that.

Justin Domingus:
Yeah, I think really containers kind of are the biggest win on this journey because you've got, as you said, lots of different applications written in different languages, different stacks, potentially really, really old, crusty, but they have to run somehow, and you need compute to keep them going, so, really, a container is just kind of giving you a standard deployment artifact that you can put anything in. So, it doesn't matter if it's a Java app, Python, Ruby, whatever. You're just building a binary artifact, that you can publish in a standard way and then run that in a standard way. So, it's actually, I think, perfect for multi-cloud and on-prem hybrid environments, because you need a standard runtime when you want to run in lots of different places. And I think that's another great thing that Kubernetes offers is it is cloud-agnostic. You can run Kubernetes on Amazon. It's going to be roughly the same as on Google. It's going to be the same roughly as bare metal that you provision yourself. So, it's just kind of this universal platform, and once you've got a container, you can run it anywhere.

David Linthicum:
John, do you agree?

John Arundel:
Yes, absolutely. I think that's well said, and it also means, you know, previously, companies that wanted to run IT operations like this had the choice of, well, you can buy a load of machines, and you can build your own data centers, or rent space in someone else's, and you can have an army of engineers to manage them all and so forth. And it's – that's not much of a choice. So, with the cloud now, at least you can say, "Well, I don't have to worry about machines; I can just buy as much compute resource as I need and network resource and so on." But with Kubernetes, the point is, you know, as Justin said, there is now this standard platform that any provider can offer, and all major cloud providers do offer that. And it means – it's not just – the point is Kubernetes is not necessarily the platform that you run on directly. You can build platforms on top of it. So, it's a fantastic toolkit for making platform as a service type offerings. And you'll notice that most of the major platform as a service offerings have now retooled to use Kubernetes under the hood rather than their own proprietary systems because they can see the way the wind is blowing too.

David Linthicum:
So, Justin, what about automated testing in Kubernetes and DevOps? What needs to change in order to accommodate that, or does anything need to change?

Justin Domingus:
Well, that is a good question, and I think when you start to think about deploying apps, testing is often neglected, or people forget that you have this big complicated testing pipeline that you still need to run, and so it's like I figured out how to run my container, but I have all this (inaudible) or other stuff involved in testing it. And so that's going to be a chunk of work. And the nice thing with deploying in containers is you can actually test your code in the same container. So, as you build new code, generally, the pipeline that we see, and that we've built places where I've worked, is, as a developer pushes code to central source control, something wakes up and automatically builds a new container with that new code. And then immediately, once the container is built, it runs the tests inside that container. So, a lot of times, you have dependencies like databases or (inaudible) or you need to download some files for testing or whatever, so that's where I think people tend to start gluing things together with, again, containers.

So what we've done, we do have some apps that require databases to run their tests, for example, so what we do is actually spawn a test database right beside the application as soon as we've built it, run the test suite and then we throw away the database, we throw away the test container, and then if the test passed, we publish that into the repository. So, there's a little – there's still, I think, a little bit of duct tape and a little bit of glue that people have to come up with, just like they had to do in whatever they're on now. But, again, I think containers can just give you that nice consistency so that from end to end, from potentially even a development work phase into the testing phase and then into the production runtime phase, it's all the same binary.

David Linthicum:
So going forward, one of the things that we're having to deal with is, in essence, looking at optimization and performance around containers, and I know Kubernetes solves a lot of problems, but is it really a matter that we're looking to leverage the platform to deal with performance issues, or is this kind of an architecture and coding issue at the end of the day? I have my own biases, but I'm going to hold them back now until you guys answer. And, John, I'd love to get response from you on that one.

John Arundel:
Yeah, well I mean, you know, I think performance is an evergreen issue. It certainly doesn't go away just because you have Kubernetes. It's not necessarily a problem that everybody has. Most people would love to have a scaling bottleneck, because they have so much traffic that the app is no longer responsive, but that's just not yet an issue for many companies, and they may – you know, you don't want to spend money on scaling before you need to. So first advice I give a lot of clients is let's actually look at how much scale you have and what kind of scale you want to get to and put in place the capacity that you need to handle that, and that may mean migrating to Kubernetes and then we can say the nice thing is, once you've done that, then scaling to very large scale becomes much easier. The first problem is getting from one instance to two. And then beyond that, it's easy to have 100 or 1,000 instances.

David Linthicum:
So, Justin, migrating to the cloud in Kubernetes, if I have poorly-written, poorly-designed applications that don't perform very well on my bare metal systems on premise, and I move them into the cloud, and relocate them into containers, and redo them in Kubernetes, what are the steps to kind of optimize them for performance to correct some of those issues? Because I know it's not just lift and shift kind of migration and wrapping in containers, or maybe it is. You guys tell me if that's the case. You really need to kind of think through how you're re-architecting the applications, how they're going to be deployed in a container model, and then how they're going to run effectively and efficiently in Kubernetes.

Justin Domingus:
Yeah, I think – I mean, that's a great point, and I think my first recommendation is probably start with lift and shift. It's like if you've got a big application, monolithic or whatever, and you want to try it out, I would say don't try to start breaking up the application first and then put in a container. I would say just throw everything you've got in there, deploy it as if it was a VM or whatever, and see how it does. Like I think a lot of times people are surprised how quick containers start up out of the box, just because they're Linux process, so a lot of times, you may see the app perform better without doing anything. But then you're right, you want to start probably thinking about how can I break this up and make it a little bit leaner? And containers, I think, generally, what you see people do is kind of break up the phases of your build, and then that way you can prune out the things that you don't need when the app actually goes into production to run.

So, an example of this would be like in Go, the Go language. You have to install Go and then a bunch of dependency management software, and then you need to go download and build a bunch of binaries of all your external modules, and then you end up with your final runtime binary. So that's the only thing you actually want in your final container. And, so, at least Docker has this nice workflow where you can kind of break those steps up, so you bring in all the dependencies, build them or whatever, and then you output that final runtime and put that in your container. So, your final runtime is actually incredibly small and lean. So, I think that type of workflow. It can lead to huge performance increases, but it is a lot of work, and I think you have to do it incrementally. So, it's probably better off, I think, to start just lift and shift, see how it does and then start breaking things up as you can.

John Arundel:
Yeah, that's absolutely right, and this is one of the things we wanted to tackle in the book, is actually fill in some of these gaps about how do you do that. How do you go right from your first line of code to having a running service in Kubernetes, at scale? And, so, we actually do this. We develop a demo app all the way through the book, starting with just Docker running it on your own laptop and then running it on the Kubernetes cluster and all the things that you can do with that. So, we figure everything that you're going to do with this is going to be true for your real app, however complicated that is. So, we start with the most simple possible thing, which is hello world, and we run that. And then the point is you can take all of the code examples that we give in the book and plug your real app into it, and that will get you up and running. Then you can start looking at the architecture issues and so on. You know, actual code in there. It's not a hand wave kind of book.

David Linthicum:
I've got one question, and it's actually difficult for me to answer, and I'd love to get your perspective on this. I mean, we just talked about what you need to do to migrate into the cloud, and some of the roadblocks you need to get around, and some of the things that are probably going to be a little bit more unmanageable in the process, and some of the testing stuff. But the thing is people come down to how much money can they make from this when you're dealing with large companies, even small companies, and probably more so small companies. So, what is the ROI from the additional investment that will be made if I'm building a DevOps team and they're going to be leveraging containers, Kubernetes, things like that? We're going to start migrating existing legacy applications over to a multi-cloud environment, leveraging the open extractions that containers are able to provide, and they look you in the eye and they go, "How much money can I make from this in the next five years?" John, you're laughing, so I'm going to go ahead and go to you first.

John Arundel:

Yeah, that's an excellent question. That's definitely the question you should be asking. And the answer to that kind of depends which way you go. So, a lot of people that we talk to are very focused on this idea that you've got to build your own Kubernetes. It's just another thing in your stack which you run on your virtual machines and so forth. And we think that actually could be a big mistake, because the fact is this stuff is very complicated. We're not bashing on Kubernetes. It's just that doing it complicates the job. It does a great deal. So, there's a lot to set up. You've got to cluster machines, and to do it securely, you've got to manage certificates and roles, and access control, and all of this kind of stuff. There's a huge amount of stuff to do, and you only have to get one thing wrong and you have a problem with your cluster, which could be a disaster. So, you could either say, "Well, I'm going to hire hundreds of engineers, put them all through PhDs in Kubernetes and then they're going to spend five years building this amazing platform for us. Or we'll just rent it from someone else. We'll just use GKE or (inaudible) or other cloud providers are available."

But the point is, because this is a standardized thing, you can rent the cloud-based offering of Kubernetes and it will be very good. You can run conformance tests and vulnerability tests on it to make sure that it's secure, and you basically click a button in the web browser and you have your cluster up and running, ready to deploy it. And I've seen figures bandied around for building your own Kubernetes center for like $1 million of engineering time. It sounds crazy, but when you think about it, it does start to add up. Your top engineers in your company are going to be spending a lot of time building this stuff, so you have to figure their salary into it, and also the opportunity cost of the stuff that they're not doing. They should be working on your business, not working on Kubernetes.

David Linthicum:
So, Justin, I'm going to go to you for kind of the same ROI question. I mean, the reality is that we've chased a lot of windmills over the years in terms of moving into different technologies, client server, movement to the web, distributed computing, distributed objects, things like that, and here we are with, in essence, container-based systems and finally getting some things right that I think we got wrong lots of times in the past. So, what's different about this? And this is a question coming from business, not necessarily me, a technologist.

Justin Domingus:
Yeah, I mean, it should be the forefront of everybody's mind is like what's the value of all this stuff, because if it's just another shiny blog post, then it's not worth doing, because it is a significant amount of effort. I think to kind of follow up with your original question to John, I think there's kind of two areas where you can see big improvement. One is if you're in a hyper growth stage where your application is just growing like crazy and you're getting tons of traffic, and you need to figure out how to scale it quickly, and it's constantly falling over because you've got so many people using your application, that's a very hard system to build yourself to handle that load and respond. So, something like Kubernetes, whether you're running it yourself or on the cloud, since you've got vertical and horizontal scaling capabilities kind of baked in, it's a no-brainer when you're leveraging that. If you're not experiencing that crazy hyper growth, your app's just kind of steady, or you're a big stable organization, I think the other big benefit is kind of what John was saying, it's just the amount of time that you save by choosing a standard platform of your operations developer people.

You know, if you have tons of your business working on your internal tooling, that's a huge cost you're just kind of pulling forward all the time. So, whether or not you can outsource that, like using the cloud, or just picking a standard platform that your team is not maintaining, I think there's huge cost savings in the long run because you're benefitting from the open source community and the work that they're doing. If they're going to be building the future and patching something like Kubernetes, that's tons of time your team doesn't have to do. So, I think it is the long-term payoff, but I also think you can't ignore that there's an upfront cost of doing the migration and doing that work. So, I personally think it pays off in the long run, but it's absolutely right to ask the question of, what is the ROI, when do we expect to see this thing pay off, and how do we know if we're on the right track.

David Linthicum:
Yeah, I get it all the time. So, John, where can we preorder the book?

John Arundel:
It's available from all good outlets and will be published I think March 31st. It won't actually be shipping, but you can preorder now at your favorite online retailer.

David Linthicum:
And where can we find your blogs and where can we find you guys on the web and reach out to you directly?

John Arundel:
So, I'm @Bitfield on Twitter, and you can usually find some of my interesting and controversial opinions there. And we have a blog for the book, which is CloudNativeDevOpsBlog.com, and we're posting excerpts from the book, plus our thoughts about various questions. And the idea is, really, we'd like to open up a bit of a bigger discussion. It's not a case of us – we've written all the answers into the book and that's it forevermore, just read our wisdom. That's not it at all. What we're saying is we would like to be asking the right questions. We have some answers, just as we said earlier, you know, we've been through this process through various companies and apps. So, we know something about it, but other people know things that we don't and so on, so we would like to try and start gathering some of this wisdom into one place.

David Linthicum:
Justin, where can we find you on the web?

Justin Domingus:

I do have a Twitter. I'm Justin Domingus, I think. I don't use it very often. I'm not very active on social media, but if someone reaches out to me there, I can probably get the response.

David Linthicum:
Not as controversial as John, just kind of posting what you're having for breakfast and where you're going for a run that day?

Justin Domingus:
I only follow my wife actually, who's a brilliant software developer. She posts much better things than me, so I figure if I just follow her and people have questions, she'll be better off to respond that I would.

David Linthicum:
That's nice. So anyway, the book is called "Cloud Native DevOps with Kubernetes," and the subtitle is "Building, Deploying and Scaling Modern Applications in the Cloud." I'm going to preorder my copy, because I think it's going to have a lot of great wisdom in it, and certainly the authors have some great wisdom. So, thank you very much, John Arundel and Justin Domingus. Thank you very much, guys. Take care.

Justin Domingus:
Thank you.