Deloitte.

Architecting the Cloud, part of the On Cloud Podcast

Mike Kavis, Managing Director, Deloitte Consulting LLP

Title: **How new practices, technologies, and players are shaking up DevOps**

Description: Even though the concept of DevOps is relatively new, there are big changes afoot in the field, and it's hard keep up with all of them. New disciplines, such as software delivery management, promise to rein in the wild-west atmosphere DevOps has spawned at some companies. Consolidation of companies and tool-sets—especially with PaaS vendors—is ramping up, and vendors are competing to rapidly develop new technologies. It's an exciting time to be in DevOps! In this episode of the podcast, Mike Kavis and guest, Sacha Labourey of CloudBees, discuss some of the latest changes in the DevOps domain and break down their potential implications for the future of software delivery—and for entrenched, dominant technologies such as Kubernetes.

Duration: **00:22:06**

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 Architecting the Cloud, part of the On Cloud Podcast, where we get real about Cloud Technology what works, what doesn't and why. Now here is your host Mike Kavis.

Mike Kavis:
Hey, everyone, and welcome back to the Architecting the Cloud Podcast where we get real about cloud technology. We discuss what's new and hot in the cloud, how to use it and why, with the people in the field that actually do the work. And speaking of people who do the work, today I'm joined by Sacha Labourey, founder of CloudBees, and of course I'm your host Mike Kavis, chief cloud architect over at Deloitte. So, Sacha, welcome to the show. Tell us a little bit about your background and about your great company, CloudBees.

Sacha Labourey:
Thank you, Mike. Thanks for hosting me. Yeah, so my name is Sacha Labourey. I live in Switzerland, and before CloudBees I was working at a company that some of your listeners might remember. It was called JBoss, one of the early companies in open source, got acquired by Red Hat. And, so, in 2010 we started a new open source-based company called CloudBees based on the Jenkins open source project, and now we are a bit more than 500 people in 19 countries.

Mike Kavis:
Wow, last time I was out at CloudBees, kind of two, three years ago, you were probably in the 150, 200 range, so you guys are growing like crazy. So, the first question – you've recently been talking a lot about a new category you guys are kind of self-defining called software delivery management. So, tell us a little bit about what that is and why it's important.

Sacha Labourey:
Yeah, well, it's fascinating to see what happens in the software space, because we keep saying that software is eating the world, every company should be a software company, and so on. And when you look at the systems in an organization, right, you have CRMs for sales and marketing, you have ERPs for finance, but you don't really have a system that makes it possible for you to see holistically how your software delivery practice is doing and how you can structure the processes across your software delivery practice, right? And, so, if you think about as a phase one as a digital transformation, you see teams starting to – or organizations start to enable some teams to become good at doing software, they do DevOps, CI/CD and so they have to break silos to achieve that.

But very quickly, if you're successful you end up with lots of teams that are successful in their own way, but they're not connected, right? Everybody has their own little systems, they have their own data, they have their own practices, and that's great. That's what you wanted in the first place. You wanted those teams to have velocity – congrats, you got that. Now the question is how do you not end up on CNN for the wrong reasons? And you know, how do you make sure you say to your bank that your front end application all goes through security testing of a specific type? You don't care when that takes place, if it's during staging or preproduction or production. You don't really care how and when; you just want to make sure that this is going to be satisfied, right, or you want to get insight into the overall behavior of your organization.

Or, as a developer, you want to know out of all of the teams working in the company who might work on a framework, what is the next best thing you can do to have the most impact on the organization? It's hard to get those insights when you're working at the micro level. As soon as you take the big picture, you get to see that, but you might end up with thousands of different signals from different systems, so you really need a system to make sense of all of that and coordinate all of that. And at CloudBees we call this category software delivery management.

Mike Kavis:
Yeah, that's a pretty cool analogy. I never thought of it that way, when you look at ERP systems, and it's kind of the ERP for delivery. That's a pretty cool analogy. I just remember back in the day I used to manage a team – and this was early Jenkins days. And, you know, this was a team that was doing stuff in C and Java so they kind of went out and built this whole delivery system, cobbling together 20 different open source projects. And there were a couple guys on the team, that was like a full-time job, just managing patching, updating that stuff. And you know, what's the value to the company, right? So, that's where this software delivery management concept to me – you know, that's plumbing, right? That's software plumbing, not so much infrastructure plumbing. And what's the value of developers spending time doing that when you could have a company that that's their core competency? So, what's your thoughts on that?

Sacha Labourey:
Yeah, I think there are two layers at which you can see that. I think SDM really applies at the higher level, meaning you could still have a company that uses different systems and different tools, and that's all fine because – well, we'll talk about the lower level. But I think the notion of SDM is really at the high level, which is you might have – actually let's take Jira, right? You might have one Jira for an organization, but you might have five of them. You might have a team use Jenkins, another team use another CI solution. But still you want to make sense of all of this holistically, so you want to define the same vocabulary, you want to define the same concept across those. You essentially want to create in some ways the data model of DevOps that's independent of the tools that people are using, because you know what? Those tools are going to change. So, that's really the layer at which we see SDM.

I think what you're referring to is more the actual factory itself, right, the tools you pick, and we're seeing definitively some level of optimization take place in organization. And we see it for specific layers, right? Typically the tools that you know make sense across your organization. So, if you think about things like Git or any code repository, right, it makes sense to use it across development or production or whatever, right? Any team might need that. It makes sense for (Inaudible) tracking or ticketing in general, right, like Jira typically. It makes sense for CI/CD because you want to orchestrate across those teams. So, those are kind of the tools that shape the backbone of what you do, but then you have a lot of what I would call local solutions.

If you do mobile development, well, you'll need special tools for mobile testing. If you do backend development, you'll need special tools for that, and so on. So, for that layer, that's where I'm pretty skeptical about the dream of unifying towards a unique solution, a wonderful stack that would do it all, because I actually think that developers are very opinionated. And they see cool new stuff – you have a cool new product coming up every month and you want to try new things. So, I think it's important to have a backbone, but then the rest is really left as an exercise to the readers, so we need to be ready for that.

Mike Kavis:
Oh, totally agree with that. Next question, we're going to talk a little bit about consolidation, and I've seen this repeat itself through history. I remember many years ago – it must've been the late nineties – BPM was a big thing. And I remember going to a BPM conference and there was, like, 30-plus vendors, and then the next year I went and there was, like, 8, right? And there's probably three now. And I see the same thing happening in your space, but what's different is your space has a lot of like systems, but they're all kind of complementary, right? So, for example, if I look at your recent acquisitions – CodeShip and Electric Cloud and Rollout – they're all similar but different, right, have overlapping capabilities. So, my question to is there's a lot of consolidation going on here in your space; where do you see this going and what does this mean for the customers as all this stuff starts to consolidate?

Sacha Labourey:

Yeah, you're right. I think that's pretty unique to the DevOps space because you indeed have "islands," right? You have the testing space, you have the development tools, you have the repositories, so you have competition within those islands, but you have coverage in terms of islands. And, so, what I think we're seeing right now is consolidation of islands that make sense together, and obviously you are going to have different strategies depending on the company. Some company you think that some aggregates make more sense than others. In our case, we acquired Electric Cloud and Rollout. It's because along with what we've built around CI and Jenkins, having a strong ARO layer for what I call the last mile and Rollout to do feature flag and some – it all is part of the same concept of I want to go from a source code to bits, and I want to be able to orchestrate that in the best way.

So, it's very clear there is this early phase of consolidation taking place today. You have a number of companies, I would say half a dozen companies, that are essentially pure DevOps companies, right? And they are starting as big magnets to start attracting some smaller, what I would call feature player companies, right, companies that have spent this amount of money and time developing very talented solutions, very nice solutions, but on their own they're hard to go to market. And, so, we're seeing those companies join the mother ship and be part of something bigger. It's really an exciting time for customers, because it really makes it possible for you to benefit from more sophisticated solutions without having to – as you were saying before, you know, to do too much of that work yourself.

And yes, so the consolidation is likely going to happen. You talked about the BPM solution. We could talk about the app server space. I was there with JBoss, obviously. All of those spaces are so important that there is a lot of pressure, right? Everybody wants to have the relationship with developers. Developers are the one having an impact in this new software era. And, so, yes, we're going to have a big fight around getting access to those developers.

Mike Kavis:
Yeah, the other thing I see here is a lot of talent being acquired. I actually know a lot of the people at CodeShip and Electric Cloud, and you know, it's hard to hire talent these days that have years of experience in this space. And part of this acquisition strategy must also be talent hire.

Sacha Labourey:
Yeah, it is. Even so, I think it's a risky proposal, frankly, because acquisitions are hard, all acquisitions. I did quite a few at JBoss and JBoss got acquired by Red Hat. We did quite a few at CloudBees, maybe five or six, and it's a big human transition. You are leaving a place where you had your heart. You were used to specific processes, and so on. And it's not that one is better than the other; it's just different. And, so, I think you'd better find – if you really want to find talented people, it's to show that your company can do exciting things, you can have an impact there, and so on. But I would not necessarily suggest acquisition as the first way to get there.

Mike Kavis:
So, in one of your keynotes – I think it was last year at your conference. I'm not going to steal the thunder in this because I can't explain it as well as you did, but you threw out this term, "Frankenstein Jenkins," which I got a kick out of. So, talk about how companies get to that state and how you guys help avoid that problem.

Sacha Labourey:
It's interesting to see when we meet with Jenkins users, there's always a fraction of the users that come to complain that Jenkins is complicated, you know, it's a big thing, we don't update it, and it's fragile, and they're almost afraid of it, and so that's not great. They like it for what it does – they like it for what it did, right, how it started. But then the sob story sometimes turns a bit sour, and why is that? And the reason is pretty simple. It's something we see quite a bit but that we should not see. That's the reality. What we see in lots or organizations, essentially, is you have the Jenkins hero, the guy or the gal who brings Jenkins into the company and is – or she's trying something pretty simple initially, just initiate CI for a team. It goes fine. It starts to automate things and it's pretty nice. So, somebody else comes and says, "Hey, that's really nice. Can I hop in with my own project?" Sure, come in. And, so, you see this aggregation of projects coming in around Jenkins, and every project is slightly different, right?

The first project is a Java project, so you put Maven and a couple of JVMs. And the next project is a mobile application, so you put the specific plugins for iPhone and Android. And the next one is a .Net project, so you get a couple of .Net plugins and so on. And there is no limit, right? You have 1,500 or more than 1,500 plugins available so you can do it all. And victim of this success – you start building this Frankenstein that has hundreds of users, hundreds of plugins. Nobody wants to touch it. Nobody wants to upgrade it, because if you touch it you own it. And, so, you start becoming a bit legacy, right? You don't upgrade the Jenkins core, you don't install the new plugins, you don't get the new features, and you start feeling bad about that. And, so, that's where sometimes a criticism around Jenkins comes, right, but it makes no sense. You should never go there in the first place. If you were a Linux administrator, essentially it means you would install a Linux box and you would say, "You know what? We're going to install the firewall on this one, the fire system. We're going to put our intranet system, IRC channel, and so on and so on, and it's going to be great.

You would end up exactly in the same situation, right? So, no sane person would do that, but because of this pattern of adoption, and sometimes with a low resource environment, right – developers are not known to always have access to a fancy budget – you end up with this big Frankenstein. And, so, the thing we do a lot when we help organizations is to sanitize the situation. Go back to a place where you have very small and nimble Jenkins focused on one team – one Jenkins, one team, very few plugins, which means they're very fast. If they crash for some reason, you can restart them in just seconds. If you want to update them – and we have a distribution for that that makes it very easy on a monthly basis – you do that super quickly.

Then the problem is how do you manage a fleet of those Jenkins, and that's where we help a lot of organizations. But that is the right way to do it. The alternative to that is a CI/CD system with no plugins, right? If you don't want to end up in a Frankenstein, you just create a one-size-fits-all CI/CD solution that's always going to be relatively stable, because it just does one thing, right? So, it's sometimes the unfair comparison because you end up with a CI solution that is very simple, but it's very simple, right? And, so, once you get more sophisticated and you want to add more stuff, sorry, you're stuck in your sandbox. You can't go beyond. So, you need to take Jenkins for what it's amazing at, but please treat it with respect and don't build a Frankenstein.

Mike Kavis:
Yeah. I start thinking about Young Frankenstein every time I hear that, so another article you wrote which is a topic I talk about, really about the future of PaaS, but it's called "PaaS: Between a Rock8s and a Hard Place," and rock is spelled rock with 8s on it, in other words Kubernetes. So, I've seen a lot of consolidation in PaaS as well. A lot of original PaaS players have either pivoted or don't exist anymore. You know, the big three cloud providers all have either a PaaS or PaaS-like capabilities. And then there's thing called Kubernetes, which has taken the world by fire. And then there's this battle between roll

your own Kubernetes and managed Kubernetes, so there's all kinds of stuff going on. So, I just wanted to get your perspective. First, tell us a little bit about, you know, why you wrote that article, but what's your perspective on where we're going with PaaS?

Sacha Labourey:
Yeah, we actually started out as a platform as a service vendor back in 2010, and we shifted to be purely focused on CI/CD in 2014. So, we definitely saw the ups and downs of what it means to be a platform as a service. It was before there was Kubernetes. It was before there was even Docker, before there was even OpenShift and Cloud Foundry, so it was quite a while ago. And I think we – first we need to admit that the growth we're seeing with Kubernetes is just amazing, right? I haven't seen such a phenomenon for a while. I think the only thing I can relate to, but I feel so old when I say that, is J2EE, right, where you had a bunch of different app servers left and right, and it was hard to make a choice because you felt you were locking yourself into a one-vendor strategy.

And J2EE was far from being perfect, but it really enabled an ecosystem for pretty much a decade, a multi-billion ecosystem where suddenly you could move from one vendor to another, but more importantly there was real competition on features across the market. So, I think this is a thing we can really relate to, but Kubernetes is really winning that war. What's amazing about it is that instead of trying to win this VM war – not to confuse with VMware – VM war, which is – you know, you've seen all of those layers like OpenStack and CloudStack trying to compete and become the new and free AWS and so on.

What's amazing about Kubernetes essentially is it bypassed all of that and said, "You know what? We're going to be just a layer above. You want to run OpenStack AWS, I don't care. We care about containers. It's a level above. We can use anything plus it's exciting to developers." And that was a huge revolution, so that's exciting. But what I think we're seeing on the market is we've seen on one hand, platform as a service vendors, like CloudBees once was, or OpenShift and Cloud Foundry, and they've been really focused at creating an environment for enterprise developers that was pretty exciting. On the other hand, you also had a lot of simpler alternatives, right? You see a serverless, you see a 4GL or the 4GL environments.

But you see those simplified environments, and in between you really have those PaaS players. But what's happening is that Kubernetes is becoming so mature and getting a lot of value, also developer type of value, that if you're a PaaS vendor these days, it becomes pretty hard to find the right value in between the right differentiation, right? You want to be more than Kubernetes – well, first, your customers want your PaaS to be based on Kubernetes, so you have to do that. But then you need to provide enough differentiation versus Kubernetes to justify some license cost. But on the other hand, if you try to go too far from the standard, well, you might as well be a very simple PaaS as well, right, like a serverless experience and so on.

So, I think the space in between is being very thin, plus at the same time you have cloud vendors coming up and saying, "You know what? We offer Kubernetes for free if you pay for the hardware." And, so, I think the breathing room for PaaS is becoming pretty small, but those vendors, they have lots of very smart people. So, I think it's going to be very interesting to watch the next few years what happens with the PaaS space, because the goal to help developers is genuine. We need to do that. But how much is going to be directly absorbed by Kubernetes, because people will trust and wait for Kubernetes to implement those, versus layers on top or around Kubernetes where developers will look for differentiation. I think it's going to be very important and interesting to watch. But just staying at the Kubernetes layer and trying to differentiate there is not going to be possible for long.

Mike Kavis:
Cool. Well, it's pretty cool to talk about the journey, how things change. Everything changes but everything's the same, I think, at the end of the day, as we see from the mainframe to where we are today. Something's new, people run to it, it consolidates, the next generation comes out and improves on its faults, it explodes, consolidates, next thing, next thing. The only difference is the next thing seems to be happening a lot faster these days.

Sacha Labourey:
Yeah, that's true. And hopefully developers still have fun. That's the important point, right? They still want to have an impact, so.

Mike Kavis:
Absolutely. Well, that's it for our episode today of Architecting the Cloud. I appreciate you stopping by and sharing your wisdom. You can find today's podcast and show notes. Head over to www.DeloitteCloudPodcast.com. Where can we find your writings, your talks, or where are you on Twitter so we can follow you and get more of your wisdom?

Sacha Labourey:
Yeah, so I'm blogging on the CloudBees blog most of the time, and I'm tweeting on @Sacha.Labourey. Just be careful. Sacha is S-A-C-H-A. I've seen a lot of different spellings for Sacha. And – but it's a very useful name. You can call your cat, your dog, or your son, or your daughter Sacha. It's a very flexible name, so.

Mike Kavis:
That's the advantage of being Mike. You can't mess that up.

Sacha Labourey:
Yeah, that's true.

Mike Kavis:
So, you can find more podcasts like this from me and my colleague David Linthicum just by searching for Deloitte On Cloud Podcast on iTunes or wherever you get your podcasts. I'm your host Mike Kavis. If you'd like, contact me directly at MKavis@Deloitte.com, or always you can find me @MadGreek65 on Twitter. Thanks for listening and we'll see you next time on Architecting the Cloud. Cool, thanks, appreciate it.

Sacha Labourey:
Thank you, Mike.

Operator:
Thank you for listening to Architecting the Cloud with Mike Kavis. Connect with Mike on Twitter and LinkedIn and visit the Deloitte On Cloud blog at www.deloitte.com/us/deloitte-on-cloud-blog. Be sure to rate and review the show on your favorite podcast app.

Visit the On Cloud library

www.deloitte.com/us/cloud-podcast

About Deloitte

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.