**Introducing Andy Clemenko**

Andy Clemenko is a senior solutions engineer and architect at Docker, Inc. He's also a technologist and DevOps analyst, with a focus on helping organizations make the transition from traditional development practices to a modern set of culture, tooling, and processes that increase the release frequency and quality of software. You can follow him on Twitter at @clemenko.

Viktor Farcic: I want to jump right into our discussion with the one question I'm asking everyone: what is DevOps?

What is DevOps?

Andy Clemenko: DevOps is a lifestyle. It's all about being able to adapt to new technologies, not only from a developer point of view, but also an operations point of view, while still being nimble. That's not to say DevOps is only that. There are a lot of other concepts built into it, which is why I call it a lifestyle. Beyond being able just to adapt, you've also got containers, twelve-factor apps, declarative infrastructure, and infrastructure as code. Yes, you've got all of these buzzwords around it, but at the end of the day, it's just a lifestyle. It's about being nimble, retooling, and moving forward. Viktor Farcic: So, how does Andy Clemenko fit tools into that picture? Because, in today's field, I'm finding that every tool is a DevOps tool.

DevOps Paradox

205

Andy Clemenko: To a certain extent, the tools almost don't matter, because you can hand a carpenter any hammer and they'll still be successful. Within DevOps, you give any DevOps or SRE engineer (whatever you want to call it these days) a tool—whether it's OCI, Rocket, Docker, Kube, Swarm, Jenkins, or GitLab, it doesn't matter—and they should be able to work with it. But again, it's about being nimble and openminded enough to embrace the next thing, which will look entirely different.

Viktor Farcic: Speaking of tools, I'm fascinated by containers. Do you think it's a coincidence that, as an industry, we've started talking about containers, microservices, and DevOps all at the same time? Is that pure luck or is there some relation behind it?

Andy Clemenko: I would say that it's a coincidence. Containers helped to accelerate that DevOps lifestyle adoption but, having worked on large Hadoop clusters, and having seen the DevOps methodologies with Puppet, Chef, Salt, and Ansible, what we've just done is effectively retooled and brought our tools up in abstraction layers. We're no longer orchestrating at the operating system layer. Instead, we're orchestrating at the cluster level.

"DevOps is a lifestyle. It's all about being able to adapt to new technologies, not only from a developer point of view, but also an operations point of view." —Andy Clemenko

Andy Clemenko

206

But that correlation helped accelerate the move up. It's still the same now, regardless of whether you're working in industry, government, or anywhere really. There's this idea that when you have a development team and an operations team, they throw shit over fences. That DevOps lifestyle is about bringing those two teams and their functions together. Forget teams, because one team with the ability to effect change is quicker than two teams trying to do the same thing. It's in this acceleration that I think containers play a part. Honestly, what I'm trying to say is that it's about soft skills. It's about the people, it's about the teams, and it has nothing to do with the tooling, just like how Docker, DevSecOps, and GitOps are all just buzzwords. We're going to get to a point where whatever object you're creating— whether it's a container, a VM, or a JAR, it doesn't matter—has the metadata within it that says how it should be shipped, and who should approve its life cycle. Viktor Farcic: That makes sense. Andy Clemenko: But I remember last year, during a demo at KubeCon, a practitioner-driven conference, Brendan Burns did a presentation on self-deploying images, where your object understands not only what it needs to be in order to be healthy, but where it needs to go and, who needs to approve its use and security provenance. So, now you've got an audit trail built in, and you're wrapping that object with as much embedded metadata as possible. Viktor Farcic: So, it's almost as if we are switching toward communication through code and metadata? I don't need to tell you what I want, as it's all self-contained in my artifact.

DevOps Paradox

207

Andy Clemenko: Exactly, and as a builder, or as a team building those objects, you can describe what it should do, while having the opportunity to divert it. But today, if I give you a Docker image, you can do whatever you want with it. I love the idea that, in the future, I could give you a Docker image that I could lock so that only you could run it, and thus you can't execute into it, and you can't do funny things with it. But it's also got a security provenance, so you know that somebody gave it to me and then I gave it to you—through cryptography—so there's at least an audit trail.

The next phase, at least in the way I see it, is having these objects really be, I'm not going to say self-aware, but at least have more meaningful metadata around security, provenance, and deployment. What if, instead of having a docker run command that was word wrapped three times with passing in volumes and stuff, you just did docker run, and the container itself goes, "Hey, I should have this, where is it? I should have this variable, and you haven't given it to me. Can I have it?" A more self-aware state is kind of a weird way to describe it.

Describing the company of today

Viktor Farcic: Switching gears a little, if you were to start a company today, what would it look like? How would people behave and interact with it? Andy Clemenko: I'm a big fan of smaller companies, where the lines between teams are blurred. So, if I'm starting a start-up, I want to make sure that our internal IT understands our product, and that everybody can work collabo

Andy Clemenko

208

ratively. I think once you start getting over a couple of hundred people in size, that's when the fences immediately go up. Something I hear time and time again from customer interaction is, "Oh, that's the networking team. They'll get to it when they can." With these fences, you have different North Stars, different goals, or you have different strategies or managers. I'm a fan of a flat organization with cross-functional teams. Like today, you might be interested in monitoring and helping with a customer solution, but that doesn't mean that internal IT can't take advantage of it. Viktor Farcic: But are these fences inevitable, then, or are they just more familiar? I wonder myself because I'm yet to see a big company that works like that, which is something I would love to see. Andy Clemenko: I think you get pockets, but unfortunately the counter to that cross-functional team is organizational stability. Because, if you've got a team, you'll find that, as your company grows, you're going to have pockets of these teams. So, the question arises about how you organize them? For lack of a better term, how do you control them, and how do you make sure that they're all moving together? The way you do that is you basically give each team a North Star, which starts to create those vertical fences. The thing with that is that it's just organizationally tough, and the problem is a lot of people end up in middle management. Because of that, there's a vested interest in keeping middle management alive. Look at it from the perspective of a 300-employee company threshold. One threshold is 100, the

DevOps Paradox

209

second is 300, and then it's about 500 to 600 or possibly even closer to 1,000. But for me, in my ideal company, I like staying in the range of a couple of hundred employees. Case in point is that I got an email last night saying, "Hey, I know you're in Raleigh next Wednesday. Can you be in Houston on Thursday?" I replied saying that I'm up for it; as long as they approved my travel requests, I'd be there, and I'd get it done. It's not my team, not my region, but they need help, so let's go. Viktor Farcic: That's dedication!

Personalities, honesty, and breathing the environment

Andy Clemenko: The other thing is that right now there are two types of personalities in all industries. It's either type A or type B, quite literally. Those As are going to go in and do what it takes to get the job done. To use an overused term, for As, it's "mission, mission, mission." Meanwhile, type Bs are, to a certain extent, going to sit back and just push the button. I see it in all walks of life. I'm a volunteer firefighter on the side, and I see it in the fire service; I see it in corporate, and I see it in government. In fact, I see it everywhere. The trick is that if you really want to keep that cross-functional team and culture going, you need to find those people that are willing to go the extra mile. Not every day, because that gets out of control. But find those people that are willing to do it, that show gumption and go do it, and then worry about complaining about it or getting compensation later.

Andy Clemenko

210

Viktor Farcic: That's very interesting, because I've had conversations with people who have said, "Oh, the company where I work is growing, and as we're growing, I'm starting to question whether I'm going to move on to something else, for the same precise reasons." I then often get a follow-up question along the lines of, "Oh, but if you grow to 1,000, that's great because more people will equate that growth with better business and stuff like that." I never really understood that because then you have to ask, what's in it for me? It's not my company. Why is it better if we are 1,000 rather than 200? Andy Clemenko: If you're looking at it from a purely financial point of view, if there are two companies, one with 10 employees and another with 1,000 employees, who is making the most money? The answer is the person at the top. So, the bigger the company, the more revenue there is, and the more the stock's worth. Are you directly incentivized to do it? At the end of the day, is money really your incentive? I wear a hoodie, and I'm an engineer with a degree in engineering who wants to solve problems and build some cool stuff; that's literally it. I'm in a place now where I help customers to solve problems and build cool

"Those As are going to go in and do what it takes to get the job done. To use an overused term, for As, it's 'mission, mission, mission.' Meanwhile, type Bs are, to a certain extent, going to sit back and just push the button. I see it in all walks of life." —Andy Clemenko

DevOps Paradox

211

stuff—I'm helping, and I love it. Do I see an extra dime if we sell an extra widget? Not directly. Maybe indirectly, at the end of the year. But that's not my personal North Star. I think it takes a certain kind of CEO to pump the brakes and not assume that massive expansion is going to solve all of the problems. Because, in my mind, not all growth is good. Viktor Farcic: I guess it depends on what you're after. I feel the same in that I'm definitely after money, up to a point. I cannot live on 100 bucks a month; but there is a limit that I reach where I'm kind of like, "OK, it doesn't really make a difference anymore," unless I've got ambitions to buy a chopper or something like that.

Finding your North Star

Andy Clemenko: It's your North Star! Putting the brakes on this interview, I want to ask what do you see? I know our discussion has focused on me, but what do you see in terms of company size and embracing DevOps? Viktor Farcic: In regard to company size, I feel similar to you in that the bigger the company gets, the less fun I have working in it. Andy Clemenko: It's great that you see things the same way as me. Viktor Farcic: I think that's kind of my definition. I feel that being in software engineering is, in a way, a privilege. My feeling for that is because we are one of the very few professions that we usually join for fun and can continue having fun.

Andy Clemenko

212

At the end of the day, as long as I'm having fun, it's excellent. It's just that I feel that the bigger we are, the less fun I have. I visit a lot of companies where I feel there's no hope. I work with them for a short period of time and show them how to do this and that. But then, I'll come back a year later and ask them what they're doing, and then they ask me, "What do you mean, 'Tell you what we're doing?' You were here last year; you know what we're doing!" Andy Clemenko: That's the thing—nothing's changed. In terms of buzz phrases or buzzwords, bureaucracy is the anti-pattern to both DevOps and the DevOps lifestyle. I just want to do the DevOps lifestyle equation, but there's really a need for bureaucracy in these big organizations because you have to be able to organize that many people at some level. Otherwise, it's going to be the Wild West. You've got to be a better start-up. I really think we need to break up those big companies and keep them small. A CEO would have to have the courage not to grow to 10,000 employees because when you do, you're going to lose nimbleness, and the ability to adapt not only to this lifestyle but also, as the wind changes, to any new North Star that comes out. But, unfortunately, money is power. What we need is the money that the big companies have in order to fund the little guys. It's like this weird symbiotic relationship that's not mutually beneficial; there's a gap somewhere. I'm on a contract right now that's 1,200 hours in, or 50 days, and our team has literally spent 500 hours of that time over the last two months getting our laptops and saying, "Hey, we need an NFS share; we need Windows VMs." We're very much in a state of saying

DevOps Paradox

213

we need this and that. The issue is that the company's response is, "Yeah; it's coming, man. Let's investigate." I've got a laptop here for them, which is always on VPN. Cool, that works great, but all of a sudden, I can't SSH into Linux boxes, and then they're blaming us for turning stuff off. I mean, I can bounce and jump through—I'm a geek—but this is clearly a firewall issue. So, then the natural response is, "Well, we'll open a ticket." Fine, but now you've got to wait six weeks for the networking team to get around to it. I'd be there saying to the networking team, "Hey, guys, do you want this project to be successful?" To which the company responds, "OK, we'll accept your million-dollar check, but now our employees are getting frustrated and annoyed because we're not doing anything." Viktor Farcic: But before I had a feeling that when I'm in those situations, it's like, you're not wasting my time because I'm getting paid for this, but you're completely wasting your money. At the end of the day, I get paid, so I don't care. But then I came to realize that maybe the perspectives are different. Actually, what I consider completely irrelevant—zero improvement—is a big deal. Andy Clemenko: I guess it's about the DevOps lifestyle, and I think it's also about moving forward. It's about taking a step, and even if it's a tiny step that went from three months to two months, that's still a step forward. Spiritually, I feel frustrated when I'm not moving forward, whether it's with a company, life, financials, or whatever. I like that forward movement. I do believe that there's a certain extent where companies feel good

Andy Clemenko

214

about at least moving forward, even though it's not where you and I ideally would like to get them.

One of the things I do when I start an engagement is try to establish a North Star, whether it's a short-term, mid-term, or long-term project. It could be a bunch of North Stars, or it could be a series, but at least you know where you ultimately want to go. Because, that way, at any point in time, you can ask yourself, "Am I in line or am I perpendicular? If I'm perpendicular, what's the cause of that?" Because sometimes you have to go back to find a new path, and that's fine, but you have to understand at least that you are going backward, away from your ultimate goal. Unfortunately, the issue is that some of these companies just say, "We want DevOps." That's their goal, but you're there thinking about how they don't understand what DevOps actually is. My favorite is when companies say that they want Docker, which is something they say all the time. But the question is what does Docker mean to them? I joke about the Docker lifestyle because Docker is not just containers. It's CI/CD. It's version control. Some of these places don't have sustained version control either through monitoring or logging. It's ELK and Splunk and Prometheus and Grafana.

"The issue is that, some of these companies just say, 'We want DevOps.' That's their goal, but you're there thinking about how they don't understand what DevOps actually is." —Andy Clemenko

DevOps Paradox

215

It's all about these aggregate systems that you bolt on to your infrastructure. In fact, it's even a little bit of Puppet or Ansible. It's understanding Kubernetes YAMLs, to which all I'll say is, "Lord, help us!"

Viktor Farcic: Exactly!

Understanding what you're buying

Andy Clemenko: But it's also Jenkins, GitLab, and all of these things. Take the project I'm on now, for example. We need version control, and we need a CI system. So, I asked the client, "What have you got?" They're like, "Well, this team over there has—" I ask, "Do you have a central?" They respond by saying, "No, we don't have a central." They may then ask, "But can we stand up on our own?" But that's not really their job. What's going to end up happening then is you'll need to go to another team and ask them, "Do you understand what you're buying?" A classic example is that you buy a car and drive it off a lot, but 200 miles later, you scratch your head because the vehicle's stopped working. You didn't realize you have to put gas into it, or that you have to change the tires, put oil into it, and clean the car, along with the rest of the maintenance. You may think of just going back and getting another car. But no, you've got to understand what you're buying.

Viktor Farcic: Exactly. I feel like one of the significant difficulties I have is that when I'm with a customer—let's say their goal is a continuous delivery pipeline—I feel that I shouldn't

Andy Clemenko

216

cheat them and that maybe I should tell them that they should not pursue, in this case, continuous delivery.

Andy Clemenko: I've had specific conversations with customers and have said something along those lines, that maybe containers aren't the right thing for them. If they're not willing to build a CI system or version control, and subsequently they're not willing to understand all of these things that make up the DevOps lifestyle, then maybe it's not the right thing for them. It comes off sometimes the wrong way, but I pride myself on being honest to my customers and saying, "Look. You're going to need this, this, this, and this." In fact, I did it yesterday at an integrator. I wrote a laundry list on the board of what they need to provide because they're building a reference architecture—infrastructure, monitoring, logging, and CI/CD—and they're coming at it from a dev angle, so they're more worried about CI/CD, but I'm telling them that providing CI/CD is only one thing, because, hey, you're building awesome widgets, but where do they go? How are they executed? It's not useful if you can't deploy it efficiently.

Viktor Farcic: But sometimes, I don't think it's only to do with willingness or even ability.

Andy Clemenko: If your goal is to do the bare minimum, then keep doing that. Likewise, if that's working for you, great. But just don't get in the way of those who want to make a change and move forward.

DevOps Paradox

217

You've got to be brave enough to say to those people that maybe you should just stay in the past. Perhaps you should just stick with Windows Server 2003 and not worry about containers, DevOps, and CI/CD, because these are lifestyles. Customers don't necessarily like the truth all of the time, but I would rather be honest with my customers up front and not try to manipulate them. I think honesty creates a healthier relationship because it establishes long-term trust, and sometimes, it has facilitated change within a customer. Every once in a while, a slap in the face might not be a bad idea. Viktor Farcic: Absolutely, at least for an academic or salesperson. Andy Clemenko: I was on a sales call yesterday, and it was just "sell, sell, sell." All this company cared about was moving forward. So, the question is, which Docker engine should they use for their Jenkins server? I feel it comes down to the fact of asking yourself whether you absolutely need the support. Is your corporate policy such that you absolutely have to have support? Because if it is, then we can just sell you two licenses for nodes and that's $1,500 a node per year. It's so tiny that it's like a rounding error for most of their budgets. My response is you could run CE, and the amount of support you're actually going to need is going to be pretty much zero because I build CE with CI systems all of the time. The company's response was for us to send them a quote. The downside is we couldn't sell professional services, including the full product suite. But you know what? At the end of the day, at least the customer feels they've gotten an honest answer from the sales guy and me.

Andy Clemenko

218

Viktor Farcic: Earlier, you mentioned Kubernetes YAML. In fact, I believe you said, "Lord, help us!" Why do you say that exactly?

On Kubernetes, Docker, and lowering the barrier to entry

Andy Clemenko: Any time there's a new technology, developers have to lower the barrier to entry, especially for changing. For changing abstraction views and for changing tooling, you've got to make it easy. Rancher did a fantastic job of making orchestration easy. They had to catalog, and my God, it was great. I had a company director once who wasn't a computer geek at all. To be able to deploy a ghost blog server by clicking two buttons blew his mind. You just have to make that barrier to entry really low. The problem I see with Kubernetes right now is that the YAML in itself uses spec four times in a single object type. YAML format is fine, and everyone can do the vertical lines and, in their code, get the spacing right. But its overall structure? Well, a customer yesterday was talking about Swarm versus Kubernetes, and how you can

"Customers don't necessarily like the truth all of the time, but I would rather be honest with my customers up front and not try to manipulate them. I think honesty creates a healthier relationship because it establishes long-term trust, and sometimes, it has facilitated change within a customer." —Andy Clemenko

DevOps Paradox

219

take a single object in Swarm, and it describes the ingress URL-FQDN, it represents the number of replicas, and it represents the number of ports and the volumes in it and one object— in Kubernetes speak, that's seven. That gets a little frustrating; not to mention that right now there are 37 top-level objects in Kubernetes. And then there's my favorite one known as the CRD, the custom one. If our theories are good enough for you, you can make one of your own, and we'll just work with it. Kelsey Hightower said that Kubernetes is not the endgame. Somebody needs to come along, and I'll tip the hat to IBM and Red Hat that OpenShift became an opinionated Kubernetes. That's cool, but that's not Kubernetes, and I think it's unfair for them to sell it as Kubernetes. Viktor Farcic: Right, so, in your opinion, what needs to come along to address that? Andy Clemenko: Someone needs to come along and really say that we're all going to use Kubernetes underneath. We understand the Kubernetes YAML, but we're going to simplify it and make our own converter app to format on top of it. That'll translate to the lower-level primitives, to the 37 top-level objects, such that the developer just says, "Here's my image," or better than that, "We talked about the metadata being transient with the image, but here's my image. Here's the number of replicas, here's the network it should be on, and here are the ports it's listening on—the number, and very simply, within 5-20 lines, it's minimal." Look at Helm: they've been trying to do that, but Helm in itself is complicated. You've got to it pull charts. I'm not even looking

Andy Clemenko

220

at Helm, and people are saying that Helm's easy. But, no, it's not. You see it time and time again as you help these companies to understand the DevOps lifestyle—these tools are wicked hard. Viktor Farcic: It's easy until it doesn't do exactly what you want, and then it becomes a nightmare. Andy Clemenko: Look at the hype cycle around Kubernetes. I've got customers that are saying, "We want Kubernetes!" To which I say, "Are you doing something specific? Are you pulling? Why specifically do you need Kubernetes?" This is a question they can't answer because they don't have an answer to it. It really comes down to somebody up high having seen it in CIO Weekly, or it's the buzzword right now, and they've got to have it. Then you actually start showing them that YAML, or the fact that in order to tie an ingress controller to a service that sits in front of a deployment, you have to have an ingress object. That's four objects now. Viktor Farcic: The reason why I'm asking is that when I jumped into Docker, I felt that it was one of the very few technology tools that I can use for everybody in a company. If you're a tester, then it's useful for you, and if you're a developer, then it's also useful for you, just as if you're an operator. At that time, Docker was almost a communication tool. It's useful for everybody, and the entry point is easy. I can explain it to my mother. But then along comes Kubernetes, which I admire because Kubernetes is extremely powerful and extensible, and it allows you to do anything, including make coffee. But now, I'm not actually able to explain what Kubernetes is anymore unless a person decides to dedicate their life to Kubernetes.

DevOps Paradox

221

Andy Clemenko: It's a religion. Viktor Farcic: Because of the complexity associated with it, I feel Kubernetes cannot be just another tool in your toolbelt. You need to be dedicated to it. So, in my book, it's useless for developers because they're never going to learn whatever they need to learn for Kubernetes. Though maybe I'm a bit pessimistic.

Andy Clemenko: No, I agree with you, because that's something I see too. The exciting thing for us at Docker is the fact that Solomon Hykes didn't invent containers; let's be honest. We've had zones, we've had attributed, and we've had encapsulation technologies in the past. All that he and his team were able to do was merely make Docker run in a simpler form, and that, to me, was the pivotal moment. I really think what we need is an operational platform—a framework—to be simple, and that's why I'm excited about Kubernetes being implemented into Docker Enterprise. If only we could take an Apple-like approach to it: let's make it simple; let's make it work, and let's lower that barrier to entry and move forward, then, hopefully, we can abstract on top of Kube just enough. Leave the door open if somebody wants to look behind and use kubectl all day long, but abstract it just

"Solomon Hykes didn't invent containers; let's be honest. […] All that he and his team were able to do was merely make Docker run in a simpler form, and that, to me, was the pivotal moment." —Andy Clemenko

Andy Clemenko

222

a little bit to make it simple enough to work. When we talked about lifers versus go-getters and big companies, I think that the minute you have a barrier to entry slightly higher than an inch, it's enough to cause a lot of resistance. If you want to effect change at a company, you've got to make that resistance—the possibility of resistance—zero. I guess that's almost like a mathematical function. The closer to a zero amount of resistance you get, the higher the probability of change within the organization. Because I know when I first started looking at Docker, I saw it as a threat, at least from a sysadmin's point of view. Viktor Farcic: You really saw it as a threat? What's changed since then? Because you're now a senior solution engineer at Docker, so your initial perception must have been wrong. Andy Clemenko: At the time, I saw Docker as a threat because developers could just do things that required sysadmins. Thus, my knee-jerk reaction was that Docker is just anti-sysadmin. But that was until my first docker run. Then a lightbulb went off, and I had the epiphany of, "Holy shit! I need to go and work for this excellent company. I'm in!" But again, you've got to make the barrier to entry as low as possible. Have you ever seen a new developer's eyes when you show them that 1,700-line Kubernetes YAML to deploy Prometheus and Grafana? I did it yesterday, and their jaw hit the floor. Viktor Farcic: I know what you mean; that's a face I see all the time. People will often call me and say, "Viktor, can you help us with this and that?," or they tell me that they want to jump into Kubernetes, and after the first half an hour it's all excitement, but then the reality sets in.

DevOps Paradox

223

I think it's interesting in the context of this discussion we're having about DevOps. But I think Kubernetes fosters the creation of those roles, and sysadmins will be able to use it. What I would like to see in the future is for us, as an industry, to stop talking about Kubernetes and see that there is something on top of it that only a few people know about. I guess it's more or less what you described for Docker E. Andy Clemenko: There is a thing with kernel developers today; there will always be extreme experts at each layer, but the number of people directly interacting with that layer becomes very small. Viktor Farcic: Because you're not working with it. Andy Clemenko: Exactly, there's no need. Viktor Farcic: I'm running Mac right now while I speak to you. I don't know what's behind it, because I don't care.

Andy Clemenko: That's a good point. I believe it was Scott McNealy from Sun, who talked years ago about Sun Grid deploying and debuting SAS Grid effectively. He said that, when you plug in your hairdryer, you don't need to know about nuclear energy. You just want to plug in your hairdryer, and you just

"I honestly don't even care whether this container is OCI-compliant. At the end of the day, I just want it to work. I want it to be portable. I want it to be secure. And I want it to be easy." —Andy Clemenko

Andy Clemenko

224

want it to work. So, apply the same thing today, because I don't care what orchestrator is underneath—I honestly don't even care whether this container is OCI-compliant. At the end of the day, I just want it to work. I want it to be portable. I want it to be secure. And I want it to be easy.

Looking to the future

Viktor Farcic: So, what's next, then? Andy Clemenko: In the near future, I see serverless picking up some momentum, but I'm still waiting for serverless to be actually written into the lower-level orchestrator directly, and not as it currently is, which is as an extra layer on top. To me, serverless is just a rapid reaction scheduler, to some extent. Elias Pereira has done some really awesome stuff with OpenVAS, to the point where it's got self-autoscaling of containers because it's deploying its own Prometheus. To me, conceptually, having similar functions at multiple layers seems redundant. So, let me ask this: if we can take OpenVAS and build it into the lower orchestrator, why don't we build into right into Kube or right into Swarm? At least that way I'm advocating for a 38 top-level object. But the idea, though, is that if you have more batch processes like serverless, they can still use the same schedule. You don't need to build on top of them and add all of this extra stuff to do the same thing. My point is, I would love to see an orchestrator just be able to say, "OK. 1 through 5 are long-running; 6 and 7 are serverless." And again, we talked about that self-aware nature. What if you had a container that says, "If I haven't been used in 10 minutes, turn me off"?

DevOps Paradox

225

In that case, you don't even have to have a separate object for serverless or daemons. The thing is self-aware, and it says, "Hey, I haven't been used. Spin me down." It tells the orchestrator, "I'm not busy, so turn me off," and then when the next request comes in, the orchestrator says, "Wake up." There you go. Why not? I say let's blur those lines. Wouldn't you say that makes it simple? Let me ask you, Viktor, do you remember the moment when you did your first docker run? Viktor Farcic: That's what I'm saying. My first reaction when I was running Docker was, "OK, I started 10 minutes ago, and I already understand how it works. I don't know what's behind the scenes, but it works." Andy Clemenko: And you were able to do a docker run and see your web page, resulting in you having that lightbulb moment, which is what we need for all of the DevOps tools. That's how change is really going to happen—with these lightbulb moments. Viktor Farcic: Exactly, but going back to serverless, would you place your bets on something along the lines of what you've explained, or something similar to Lambdas with cloud proprietaries and all of those things? Andy Clemenko: I wouldn't place any bet because computing today happens everywhere. It happens on your watch, it happens in your datacenter, and it also happens in someone else's data center. There'll always be this balance between on-premises and the cloud, and serverless and full daemon, or whatever you want to call it—server/serverless. It might not be 50/50; they'll flow. I think there'll always be both because of

Andy Clemenko

226

security and financial reasons. Too many times I hear customers saying that corporate policy says we can't touch the internet, so they're fully air-gapped. You can't use Amazon, you can't use Azure, or there's a project team building a VPN to the VPC, who'll dedicate a link, and all of that good stuff. But there'll always be this balance, and, indeed, all we're doing is just shifting responsibilities. So, do I think serverless is going to take over? No, but I think it's going to consume anywhere up to 20% of the container space today. But guess what? What format on the backend is serverless? Viktor Farcic: Kubernetes? Andy Clemenko: So, it's the same underlying fundamental object, and the same construct. So, why can't we just make the construct more self-aware, whether it's a batch job—which serverless technically is—or a long-running daemon that's constantly serving traffic? Viktor Farcic: Because my current concern with serverless is that I need to choose which platform to use and then almost stick with it forever. I would technically have liked what you just described—tell me how to explain something and then tell me whether it will run as Lambda, Azure function, or VAS. But that shouldn't be my concern.

"My current concern with serverless is that I need to choose which platform to use and then almost stick with it forever." —Viktor Farcic

DevOps Paradox

227

Andy Clemenko: It shouldn't be, but for me, it's a process. Fundamentally, inside the container, it's just a process executing whatever—whether it's wrapped in a Lambda function, Azure, an OpenVAS container, or a long-running Kube container, it's still just a process. The process doesn't care what it's encapsulated in. It doesn't know that it's not a conscious being where it spins up and it goes, "I'm alive! I'm dead. I'm alive! I'm dead." It just runs. Having to build separate frameworks is creating, in my mind, more confusion. Granted, there's job security. But again, it's not a low barrier to entry; although, having played with it, OpenVAS is pretty darn slick. It's straightforward to create a function, it's straightforward to integrate it, and it's effortless to execute it, not to mention it's got autoscaling and all of these fun things. But again, I'd love to see that completely integrated with a single orchestrator. I'll give Amazon a lot of credit. I don't like what they're building, so to speak, but I'll give them a lot of credit for lowering the barrier to entry. They've made it too easy to consume databases in VMs and object stores. But if you actually dig into it, it's incredibly complicated, with CloudFormation templates and all of the IM policies and security groups. I personally don't use AWS or any of that stuff because it's too complicated and annoying.

Viktor Farcic: But when you said they made it too easy, my first thought was that it used to be, but nowadays it's not.

Andy Clemenko: Actually, you're right.

Andy Clemenko

228

Viktor Farcic: I prefer DigitalOcean now, because it has what I need, and it doesn't h ave 50,000 other things that I don't need but am encumbered with anyway. Andy Clemenko: I'm a huge DigitalOcean fan. Viktor Farcic: To be honest, I've worked a lot with AWS, and I still don't fully understand how it works. But now that I think about it, nobody does; it's just madness. I have a feeling that they went in the same trajectory as we described earlier for Kubernetes stuff, but it's becoming heavier and heavier. Andy Clemenko: Exactly, and I think one of the disservices or discredits to Amazon is that they've made AWS very sticky because of how complicated it is, and to a certain extent, Kubernetes is going down that same path. It's very sticky because once you get it, you don't want to use anything else. Just look at the fact that if you put "AWS architect" on your resume or "certified Kubernetes" on your resume, your phone will not stop ringing. That's good for the person whose resume that is, but, you know, I think it leaves a lot of the little guys out of the market to a certain extent. Viktor Farcic: But, you know, if being AWS-certified is in high demand, that means that it's actually too complicated, because I don't think that anybody says, unlike with Kubernetes, that they're a container-certified person. Andy Clemenko: Or that I'm certified in Docker and Kube. Viktor Farcic: But what do you get certified for in Docker? It only takes two days to get certified.

DevOps Paradox

229

Andy Clemenko: Our certification is at a basic understanding of registry, and push and pull, and things like that. But you can absolutely learn it and pass the exam in a week or two; it's not hard. Viktor Farcic: Exactly. Anyway, I know we're out of time now. It's been great talking to you, Andy. Thank you so much for your time.