Capgemini

CR018

The state of platform engineering with Jennifer Riggins

CR018

The state of platform engineering with Jennifer Riggins

Disclaimer: Please be aware that this transcript from the Cloud Realities podcast has been automatically generated, so errors may occur.

[00:00:00] Wait, I wanted to check. I'm pronouncing her name straight. Yeah, but mine is not very helpful when you read it. It actually makes it harder. It actually makes it harder.

Welcome to Cloud Realities, a conversation show exploring the practical and exciting alternate realities that can be unleashed through cloud driven transformation. I'm David Chapman. I'm Sjoukje Zaal, and I'm Rob Kernahan.

This week we'll be talking about platform engineering, an essential part of modern operating models.

But is it being done well today? And how does it help with managing developer cognitive load? Joining us this week is Jennifer Riggins, tech culture journalist. Welcome, Jennifer. Great to see you. Thanks for being here. Do you want to just [00:01:00] introduce yourself and just say a little bit about what you do?

Sure. I am Jennifer Riggins. I'm clearly American, but based in London. Currently, I have been a journalist or working writer since 2003 and in the tech industry and for over a decade. Now I'm aging myself and I am not technical at all. I believe that if tech is so personal that it's in our bodies, it's in our homes, we are expected to use it in work and school and at the doctor's offices.

Everyone should understand what's happening in tech and the consequences of technology and that's it. Everyone should have the option to participate in the creation of it. So you can imagine which culture side of tech I love around diversity, equity, inclusion, accessibility, ethics, everything in that area.

And then logically my beat for the new stack, which [00:02:00] I've, I'm a freelance writer, but I've written for the new stack for about eight years now and has been platform engineering because It is that important to the lives of the developers it's serving. So we're going to talk a bit about platform engineering, I think, something you've been thinking about for the last weeks and months.

Let's just start with the real basic. In your mind, what's the definition of platform engineering? How expansive is it? And where's the state of the art, if you like? Okay. To start, my definition will be probably more what it could or should be than it is actually at a lot of organizations. Yeah, of course.

It's the team or at some companies, it can be the company if they're so small, that is dedicated to the internal developer experience and they are building a layer, totally varies where in the stack and where in the company it is, that is to help guide. [00:03:00] The choices. So for a while you had all these autonomous developers and all these tooling and it's made a mess and it's very difficult to onboard and it can take hundreds of days now for developers to really be able to release something.

So platform engineering is about understanding the best practices for your organization and understanding the bottlenecks that developers are having specifically, but not exclusively around operations. It can also include security regulations, being compliant, things like that. But operations, it's usually, some companies have wittily said DevOps is dead.

No, it's just an evolution because dev was forgotten DevOps. And it's helping them to release faster. And especially now in this time of layoffs to do more with less. And it should be specifically [00:04:00] responsive to what developers are saying they need and things that at least more than one team is using.

You're not going to create these, they call it golden path or golden pathway or which is very a la Netflix's guardrails, not gates. So you can do your own thing, but then you probably won't

be supportive and be supported by the platform engineering team or the platform team. But at some point, maybe that thing you're doing.

Becomes the best practice for the organization, and then the platform team will take over helping do that. This could involve Kubernetes because Kubernetes is really complicated. It can involve so many distributed things that we've gone so far in the distributed that we need to come back and make it easier.

for developers to do their job. Rob, how in your mind does the discipline of that fit into the products and platforms operating model? And it's an excellent explanation about what a platform is really clean. For me, the product [00:05:00] focuses on the business and gets the intimacy with the business and focuses on what they need.

The platform. Creates an environment that allows that business logic to be assembled quickly in a secure, compliant, safe way that meets regulation and provides all the plumbing that actually engineers do like to get involved in, but shouldn't because certain things should be shared. Other things should be unique, but the product gives the focus to the business.

The platform gives the focus to the stability, the consistency, the efficiency, reducing toil and all those things. And it's that balance striking too, that I think. For me, brings out the importance of platform engineering, giving you harmonization and fundamental acceleration, but I realized there's something I forgot, which is the business side, which often is forgotten or likewise or in the opposite.

Is that why you did that to demonstrate why it happens in real life? Yes, absolutely. And let's say that the platform usually helps. [00:06:00] And for it to continue to get funding because platform engineering is where we're seeing these platform teams are seeing some of the major cuts right now, even though it's pretty innocent of an idea for most communities, the platform team should show transparency and Thank you.

to the business team and specifically help the developers feel connected to business value. So it really is a developer audience, but it's also about helping them understand that business value and increasing their speed to business value. So the work that a platform team. Pretty much it's not necessarily business differentiating, it's more about, I liked your phrase there, the guardrails for the organization.

So presumably that includes cyber, cloud, regulatory compliance, those sorts of things. First of all, is that right by your definition? And secondly, this is the place where policy as code gets implemented, I would have thought. Yeah, often it is usually it were infrastructures code policies cold things like that I'll be bang sir for some TASO [00:07:00] which has an open source developer internal platform engineering products called cratics She put it the best way.

I think a Infrastructure scaling security these are obviously Important things, but they're not what make a development team differentiating. They're not differentiating a business or anything. So they're, but they're not unimportant. So it's about taking away these. important but non differential tasks from the development team, so they can focus on the business differentiators and be faster at delivering them.

And in your view then, engineers and developers like to play with the plumbing and fiddle. Basically, whereas they should be thinking about value back to the business, especially in the product, where do you think that balance strikes with trying to get the business end of the product focused on creating the outcome and allowing the platform to serve?

Because, it's that if they don't have a compelling experience, the developers want to go and

build their [00:08:00] own thing because they can and they're inherently creative. As I said, there's a, is there some view of how you strike that balance between the developers want to use the platform and it serves their needs correctly?

It's interesting because the platform. team is failing if the developers going off and doing their new shiny because they're not properly solving the team. And that can happen because there is a risk specifically in platform engineering because platform engineers are first and foremost engineers.

So they also want to build the cool things. But that is not the job of the platform team. The platform team is building those best practices and the stable things. So they need to use that Hold back, but when you say they that developers like to play around with stuff and all I think the best example is last year's Linux foundation survey said that only 3 percent of developers want to be responsible for security.

Now, we also have this beautiful message that everyone should be involved in security. And we [00:09:00] also have security getting harder and harder forever and ever. And we also have the fact that it takes over a year for the average open source patch to be fixed. So compile that all together and maybe you don't want everyone in charge of security.

And I think it's that thing, the value prop to the developer will take the pain away so you don't have to worry about such things. Yeah, and then you can get on with the things you want to do, really. Exactly. Jen, you mentioned earlier on in the conversation that you were going to give us a definition that was maybe not reflected in real life.

Perhaps we can scratch the surface of that a little bit. In the companies you're talking to, what are you seeing in real life and how far away from the version of good is that? Often, as a journalist, I'm talking to the companies doing it well, so there is that, but it's just boom, we're an agile organization.

Boopity bopity boo, we are DevOps. Boopity boo, we have a platform. That doesn't mean it's actually a reality, especially if [00:10:00] it's top down. Does it need top down funding and commitment in the long term? And again, does it need then transparency to that top? To continue that funding, especially as budgets get tighter for probably no reason.

Yes, it has to be driven by the developers. So there are certain companies that are doing it in different ways. You can borrow a developer from each app or each domain and bring them onto the platform team for a couple months and have them build what they want out of it. Or you can embed platform engineering developers or platform engineers into the developer teams to figure out what they want.

You can do lunch and learns or constant demos. You treat them as a customer, treating them as the old fashioned sysadmin throw it over the wall will just turn to failure and just another. [00:11:00] Another silo. And you think that's a function of empathy? Teams don't like each other because it wasn't invented here.

But actually, with that rotation you discussed, which is quite nice about you go and build your own platform and then we'll take care of it. And then the next sequence comes in. It's it's walking a day in the other shoes. And then you get that empathy. And then suddenly, the interaction works a lot better in the organization.

Do you think that's the best way to go forward? Absolutely. They say that the biggest step for an architect or for a customer relations rep is empathy, and it's about treating your developer like their customers. They are internal customers. You should have measurements. You should have an NPS survey. You should have results that measure if they are satisfied.

And of course, if they are using, which means making it easy for them to use having documentation, Yeah. I would say more documentation than you would have for an external customer because you need to have why [00:12:00] you've made these decisions. And external customers shouldn't usually be privy to the reasoning behind architectural decisions.

But an internal customer needs to understand that. And that becomes more so true in this time of, again, senseless tech layoffs when We don't have this knowledge transfer that is necessary. I'm interested in that conversation about tensions, simplifying for the sake of the conversation, but between the vertical product towers and say the horizontal platform towers and your ideas of maybe you should take some platform engineers and put them in the vertical and maybe take some application engineers and put them into the horizontal.

What specific methods have you seen that are successful? To my mind, that's an ongoing part of the iteration and the performances is continuing to work on. that tension. So recognizing it's there. Don't just think you've got it fixed like a once and done, process that's now in place. It might be ongoing and surely impact your backlog management [00:13:00] and things like that.

So what have you seen that really works in that situation? And and perhaps what have you seen that doesn't work that well? It's platform as a product, so it's not adopting, even if you are using external sources and there are several, all the whole gamut of an entire platform, boom, you have it to backstage, which is an open source tool and things that are taking pieces together, but you have to treat it as a product with customers that you are consistently talking to.

And it can't be a one off, it's useless then because your customers, the developer needs, are going to constantly change, so you need to have the, probably a tighter feedback loop, which is harder because I would say your colleagues are harder customers than the ones that pay you directly. Yeah. I would say that also that the, I think Rob, you mentioned the word empathetic like the relationship between humans in that situation is so critical because the last thing you want to [00:14:00] do presumably is put some form of governance body in place that's like arbit, arbitrating the process.

That seems to me to be a completely retrograde step and almost gets us back to ITIL world. Absolutely. And still you have those golden paths, but you don't have to stay on them. There will be some teams that are intentionally innovative and experimental, and that's important to your entire organization.

Yeah. Although Dave, I love the idea of a theme park called Eitel World where we are in a place which is extremely process focused with lots of gates to walk through to allow us to have fun. That's my new, that's my new dream. I can imagine the merch. I can imagine the merch already.

it could be sat next to Badly Done Agile World where it's just complete bedlam and nothing works. It's where we literally have a wall. Yeah, DevOps world where everybody's climbs over walls. This actually sounds more like a gym, or the place where you don't have a lot of equipment, [00:15:00] than the happiest place on earth.

That's the tagline, the happiest place on earth. Oh, right there. Let's get back to platform, shall we? So look, sorry they've been around for a while. There's been a number of goals, hasn't they, over the course of the last decade or so, platforming things. Are we just in another one of those iterations, or do you think any of the contextual conditions have changed, Jennifer, that creates a, maybe a way to do it better or differently this time?

I have an opinion on this, which, at least Adrian Kockroft and I were butting heads on Mastodon, of all places. I've never, I've not been on Mastodon yet. It crops up on Twitter the whole, the whole time oh, it's much more friendly over on Mastodon. Very decentralized, though, isn't it? The way it operates, yeah, I don't get it. Anyway, that's probably a whole other show. Sorry Jennifer, I didn't mean to interrupt you there. I have a [00:16:00] theory that open source could require platform engineering as your cost for your free software. Because of things, like I mentioned earlier, that it takes It's hundreds of days for the average company to run a security patch, like an important security patch, because if proprietary, it doesn't yes, proprietary unto itself can be more complex.

And you have buy in, vendor lock in, which makes it harder to make changes and things. But certain things are updated, crucially, because they are publicly traded or they have a lot of PR and things like that. Versus open source, not always. We just have the biggest issues. When you have that, and when you have the complexity that is, Kubernetes, which developers, I don't think, as you mentioned, Rob, developers really want to get in there and play.

[00:17:00] I haven't heard that much delight for developers in the Kubernetes space. They'd rather have it abstracted out. So that's a very logical place to abstract out things like that. But like you said, Dave, it's not an uncommon thing to have a platform. The difference is, I would say platform engineering is a mindset and discipline.

That, like the best parts of DevOps is looking for that tight feedback loop is looking for that constant amount of information and growing and everything in between extreme programming, verification, DevOps. And it's just extending it to, ironically, the first half of the portmanteau, the developers.

And it's a good point around the platforms are a great way to control risk. And as business works out, the technology is its future. They [00:18:00] are one in the same and business get more involved. They start to look at. Technology differently and think this could wreck my brand if we get it wrong. Or if this goes horribly wrong, we won't get into that customer base or frustrate them.

So that risk balance starts to kick in and actually a platform is a great way to manage it effectively, especially when you talk about the average time to patch and how you need to be consistent and everything else. So I think that is a function of the business becoming much more interested in technology that will continue to see the rise of the concepts like platform engineering is a fundamental way to manage risk better.

Rather than just let the wild west occur with the development environment. In my mind, getting your platform right is what frees the developers off. If you want to create a situation where you want high rates of innovation, high rates of creative freedom, and high rates of experimentation, you've got to get your platform right, don't you?

It takes you from code monkeys to creative workers. Yeah, exactly. Exactly right. Exactly right. You also touched on as we went through that, I think you might, I think you might have touched on it a couple of times. And I just want to return to this [00:19:00] notion of does platform engineering mean DevOps is dead?

In my head, I'd always held them as being Part and parcel of the same high quality discipline, where were you picking up signals around DevOps being dead in the wake of platform engineering? And what's your view on it? This is, this might just be a really catchy. marketing ploy by a certain platform customer or a SAS provider, or I guess past provider, obviously platform as a service provider.

So there can be that, but it's also just excellent headlines. So I can't hate on that. But Daniel Bryant had from ambassador labs had told me or taught me about how It really is just helping the fruition of DevOps more, maybe again, concentrating on that first half of the portmanteau, and it's really grounded in that way of the three ways of DevOps still of concentrating on flow and systems thinking towards delivering end to end business value.

faster, [00:20:00] enabling those developers to add, deliver that business value faster, creating even shorter feedback loops with those developers, and then of course, continuous experimentation, learning, and improvement. If platform teams adopt DevOps, you actually make it platform engineering. Because you are treating, they have to be your customers, developers are your internal customers, they're your customers because they're the ones you owe, and your metric should be towards.

Yeah. That's the hand in glove sort of relationship that I thought they had together this natural bedfellows in my mind. The other thing that I wouldn't mind coming around to, perhaps by way of bringing our conversation today to a little bit of a close is going back to the relationship between product towers and platforms.

It seems to me that if you left. Everything that you needed to do from, metal all the way through to running interface with the customer to every single product team. Not only do you get a higher risk of looseness in your underlying security and regulation and all of those sorts of things, also unhelpful non [00:21:00] standardization across how you're developing things.

But actually, That's a hell of a lot to take on as a team, isn't it? And it's a hell of a lot to take on as a series of individuals. If for no other reason, splitting out between the disciplines and getting the disciplines focused, but also just reducing the cognitive load. It's endemic in tech.

Before COVID, it was about 62 percent of developers were burnt out, and not everyone knows how to recognize burnout. Two years into the pandemic, it was like 68%, so it's not the pandemic's fault. It's the industry's fault and how complex systems are getting. So this is a way to decomplex there, to do more with less.

It keeps coming back to that, which I don't like that term when we're talking about unnecessary downsizing and layoffs, but it is a way there is that financial opportunity that you need to show. The business or else these jobs will be cut, too. So I hate to say [00:22:00] it, but it is to do more with less platform engineering as well, and you just have to log into the CNCF framework and have a look at all the tech that they that's in that framework.

And then there's the job advert that says somebody knows everything in the CNCF framework. You just think, Who is this unicorn? It's reducing that. Yeah, it's just take the pain away. Go back to that, which is I can't cope with the complexity now. Please make my life easier. Yeah. Yeah. And not only is it the complexity that you guys have been talking about, but of course, it's the expected rate of innovation then that's supposed to come out of all of this, because you're doing it in the first place to be faster, nimble, more creative.

So you're not only taking on, vastly complex technology integration, but you're actually also have higher standards to meet in quality, higher standards to meet in innovation, right? You're leaned so far in at some point it's hardly surprising that it has a human impact. And the cloud is a huge differentiator here for the industry and for an organization to speed up.

But for an app team, [00:23:00] the cloud is something they need to speed up, but it has nothing to do with the differentiators or the business value they are driving. So why do they have to know about that? And then on the other side, it creates a really positive business

benefit that a platform may enable FinOps to start being able to track the cloud spend, which is in turn the closest proxy we have still for your environmental spend.

Yes. carbon footprint. So these are all benefits. But again, developers shouldn't care about that. Developers should care about delivering, doing whatever they need to deliver faster. And we know they, with this cognitive load, they have to go faster and faster because we as consumers, the end customers expect things like right away.

So Sjoukje what have you been [00:24:00] looking at this week? Each week I will do some research on what's trending in tech. And this week I want to focus on platform engineering in 2023, doing more with less. Look at that as if we'd planned it. Yeah. So with the introduction of microservices and cloud native development, developers needed to shift left and became fully responsible for the entire development life cycle.

So the platform engineers, they paved the way for the developers, providing them a secure platform with guardrails for their software to land on. So the developers don't need to worry about that anymore at all. We could say, because platform engineering is also building on the traditional DevOps practices, like we already mentioned before.

So you could say that platform engineering takes the spirit of Agile and DevOps and extends it with the context of a cloud native world. I think that the shift from traditional software development to developers [00:25:00] becoming fully responsible for the whole software development lifecycle does improve the quality and the time to market.

But I also think that you should not make them responsible for decision about which services to use, which clouds to host on, security and infrastructure. Because this is extremely hard to get right. And that's why it's so important for organizations to invest and in platform engineering. So my question for you, Jennifer, because you already mentioned a couple of things around this topic, but what do you think is the best way for organizations to implement platform engineering in their software development process when they are starting with this?

Ask your developers, do lunch and learns. Guess what? This is a, this is the big secret to everything in tech. People love free food. Get them to sit down, provide them with free food, provide them with the drinks they choose to be inclusive and [00:26:00] ask them what their bottlenecks are. I think Paola again was Paola Kennedy, who's the CEO of Sintasa.

They're fresh on my mind right now, but I think she was the one that said, look at JIRA tickets and to find common patterns among JIRA tickets and to understand Where are the common bottlenecks are it could be really simple. It could be something. A great way to start is documentation of how to do something.

Start as small as possible. This is one of those team topologies idea of minimum viable product. So the minimum viable product can be a wiki of how to do something. You don't need this massive beast yet, or maybe ever. It's anything that you can do, and it doesn't have to be a platform to be a platform engineering mindset, but anything you can do to smooth the pains that developers are facing in their release process.

Okay, and then one more question. I have worked with [00:27:00] several organizations that had a platform engineering team and several developer teams that were their customers, right? But what actually happened in reality that most development teams were waiting for the platform or new services on that platform to be fully finished with all the security guardrails, all the internal processes.

Do you have any tips around that? How can they speed up? Deploying and engineering that

platform. What you just described was a waterfall platform. We don't want that to happen, but definitely it does happen. They're going too big. They're becoming a sysadmin, throwing over the wall again, and saying, this is what you do.

It needs to be back to the What Manuel Pais and Matthew Skelton did with team topology needs to be a minimum viable platform. They are starting way too high level. They're starting way too strategic when the strategy is often just figure out the little things [00:28:00] first, the low hanging fruit. And security is a big one.

Security should not keep you from doing anything else, but don't start on something unless you're going to start maybe with the security first. There was that thing about actually linking the evolution of the platform to the open source model, which if the platform doesn't do it, others can build it and then pass it over to the platform to be integrated.

So it's a community development and then the platform brings it in. Does all the things it has to do to make it work. Jennifer, in your position, do you think that would work with platform engineering? It's a good strategy to allow us more community build process at the beginning to get round.

This has to do all things to all people. There is that you could do inner sourcing for sure that's another method that we didn't even talk about that platform engineering. You want this, you build it, we will integrate it into the platform, and then you have it. So that's this very similar as barring the people but maybe you have that team focus on that if it's that important for them.

[00:29:00] And then they can share it. Or maybe some team has already built a solution and then they share it. So there's that. But there's also backstage, which is Spotify's platform, which is a good starting off point in open source for this. There are varying sources, things like that. It's also like I would even this may be controversial, but I would call an API gateway a platform.

When you've got disparate sources and finding a way to discover what exists in an organization, whether you're using a tool or not, I think that is at least in the essence of platform engineering and just closing on that point, then the sort of the trust that has to exist within the organization. What has to be true for these types of strategies to work?

Because everybody has to rely on a lot of others. When you have a platform, you fundamentally rely on it to achieve your outcome. Where do you think trust fits into it all? Platform teams should definitely have an SLA and SLOs. It should have, like I said, an NPS, the Net [00:30:00] Promoter Score, or other things like that.

It should have a backlog, it should have a public roadmap internally public, because why not? So people know. But I think right now, And it's interesting because I feel like I only started really hearing about platform engineering in mass since December. But in the last month, it feels like there also has to be trust that the platform team will stay there.

Because that is one of those areas that are cutting, that are getting cut along with DEI, accessibility, ethics teams. They seem to be cutting a lot of the important things. People have worked at companies for 10 years cause they earn more than six figures or in the six figures. So let's just cut them.

But then you lose all that domain knowledge. There's a lot of lack of trust in the tech industry suddenly that's happening that we do need to reckon with and that filters down that trickles down within a company and To trust that something's going to be [00:31:00] there What if that platform is not maintained because the company removes that team and

then it becomes more crap you're relying on And it's not good anymore.

Yeah. It is the same debate around cloud sovereignty, about do I trust that the hyperscale is going to be there? And the geopolitical situation that we see, it's the same paradigm just on a different scale. It's, you've got to trust your internal platform. Like you trust your hyperscaler, I suppose You have to believe it'll be there for the future, so you can rely on it and your leadership has to trust in and invest in and buy that.

Yeah, and also know that it is not the specialization of a developer. Good. Thank you, everybody. Great conversation. Thank you, Jennifer, for your time this afternoon. It was great to see you and spend a bit of Friday afternoon with you. Sorry, I ended on a bummer, but thank you. No, it's a reality. I've got the idea for Ittleworld and I'm gonna get one of them going up in Florida quite soon.

I think that I think everything we talked about is now copyright cloud realities production. He thought of

[00:32:00] it first. Thank you again, Jennifer. Brilliant stuff. We end every episode of this show by asking our guests what they're looking forward to doing next. And that could be anything from, yes, it's the weekend in about an hour's time. Cannot wait, bring it on. It could be all the way through to something in your professional life that you're excited about doing.

Jennifer, what are you excited about doing next? I'm theoretically cautiously optimistic about spring in London because we've had a long winter, but we've had all four seasons throughout this afternoon. It's pretty bright at the moment though, isn't it? It's quite nice. So in my end of town, it's pretty dim right now in dark gray.

That's how fast the weather's been changing and maybe the rainbows that may appear because of that weather. And I am cautiously optimistic about my talk at KubeCon. I always go to KubeCon as a person covering, but I also applied and was accepted to give a talk on increasing belonging and open source communities, [00:33:00] which is a socio technical problem as everything is.

It's neat. It's fun. A lot of time talking about the socio technical problem of platform engineering. There are technical solutions, but there's people involved. And that is true when you're especially talking about overworked, underpaid, or not paid maintainers trying to cultivate. Other unpaid work, which comes with privilege to be able to do volunteer work.

So it's and we know there are so many benefits to participate in open source. So I hope I get a good crowd and I don't disappoint them and I can keep it under 35 minutes because it's getting long already. But I'm excited about helping spread the learning from dozens, if not hundreds of people I've been privileged to learn from because I don't know anything.

I'm just learning from other people, which is the best job in the world. We wish you all the luck in the world with that and look forward to hearing about it at some point. So a huge thanks to our guest this week, Jennifer, thank you so much for being on the show. Thanks to our producer Marcel, our sound and editing wizards, Ben and Louis, and of course, to all of our listeners.

We're on LinkedIn and X, Dave Chapman, Rob Kernahan, and Sjoukje Zaal. Feel free to follow or connect with us and please get in touch if you have any comments or ideas for the show. And of course, if you haven't already done that, rate and subscribe to our podcast.

See you in another reality next week

[00:34:00]

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of over 360,000 team members in more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2022 global revenues of €22 billion.

Get The Future You Want | www.capgemini.com