Capgemini

CR041

Unleashing developer creativity
with Ramiro Berrelleza,
Founder and CEO, Okteto

CR041

Unleashing developer creativity with Ramiro Berrelleza, Founder and CEO, Okteto

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

[00:00:00] Can you do a dirty sparkling water? Maybe . It doesn't sound great. No, I like it. It doesn't sound great. Yeah, that's true.

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.

And this week we're going to be talking about unleashing developer creativity and productivity through automation and AI.

Enormous gains can be created through correct set of tooling, correct platforming for developers, and we're going to explore that subject today in some detail, but before that, Rob and I talk regularly about what's going on in the world and in the world of tech. And often, as I've said before, he's walking around holding his [00:01:00] head.

Trying to get his head around something that's confusing you this week, Rob.

This week, Dave, it's people who create stuff that we just don't need. Now they say necessity is the mother of invention, but sometimes it appears that lack of necessity appears to be the mother of invention as well. I'm going to use, I'm going to use a device that I've seen and is saleable. Apparently.

An internet connected toaster and I just cannot for the life of me work out what the value proposition is of having an internet connected toaster. So what does the enablement do? You can control the toaster from an app on your phone via the internet. And I'm like, it's brilliant when I'm leaving the office and I'm almost at home, I can turn on the toaster.

I don't think I've done 50 episodes with Sjoukje [00:02:00] now, I've never seen her so excited about anything. But you're like, I get like smart home and home automation and turning the heating on as it detects you nearby and all that sort of stuff. All very good. But they're just, I just feel I'd love to have been in the production meeting of saying I have I got an idea for you. Let's create one of these. The workshop was called what can we internet enable and then they had like loads of different devices around and they're just striking stuff off and then they get down to a juicer or a toaster.

You can imagine top management coming down and saying I've heard IOT is a thing. Internet enable every one of our products. I just don't get it. I can get, I just, who needs it. And if you are a fan of internet connected toasters, please let us know. Cause I cannot work out why such a thing exists.

But anyway, that's, what's confusing me to fair confusion. Cause the big scale up of the[00:03:00] use case is so niche that you could have done it without any of the. 50 million of additional investment you had to put into something and then no one could quite work out another use case. You have a field of dreams. If you build it, they will come situation around like a giant platform.

So I think you raise a good point. And I think we've moved past that in some of the new models of organization where you people are now thinking much more in outcome terms and In use case terms, but, perhaps not in the toaster world, it's like who stands back and says, I'm glad I spent 400 on that toaster.

I just want to meet that person. Look on that note, I think we've solved that problem for you, Rob. So I think we've worked through that very clearly. And we've got to go to a solid conclusion. And let's turn to our guest today and our main subject of developer automation. I'm absolutely delighted to say that joining us this week is Ramiro Berrelleza, Founder and chief executive of Okteto.

Welcome [00:04:00] Ramiro. Great to see you. Thanks so much for joining us today. Just want to introduce yourself and tell us a little bit about Okteto . Hi everybody. Happy to be here. Thank you for having me. My name is Ramiro Berrelleza. I am as I said, founder of Okteto and Okteto is a platform that. allows software teams to automate their entire developer experience, everything from remote dev environments to cloud infra, acts to secrets, all with a focus on making developers more effective and to bring, more joy to the software development process.

Tell us about the founding. idea for Okteto and then what the first steps you took on the founding journey were, Ramiro. That's an interesting story. I've been an engineer my entire career. Before Okteto , I've worked on small startups, big tech, consulting, like everything, and one of the constant kind of pain points I've always had as a developer was just how hard it was to get to work.[00:05:00]

You have a new project. You're working with a new code base or simply like a new feature of your existing project. And just that sheer amount of time it took to get your database, get your data, run your services. It was something that always annoyed me as like, why do we have to do this every time?

Especially when I started working at some point in my career, I work at Microsoft and I was working on an Azure cloud service. And that to me was like that tipping point where we were building this cloud spanning services. And yet everything was running locally, manually configured. And that's when I started to feel like, Hey, this is not the right way to do.

So I started building some things, proto, Okteto , while at Microsoft, internally. Then moved to Atlassian, had the same challenges. So then when my co founders and I were discussing, Hey, what do we want to build? What's our next thing? It became very natural to all of us and, Hey, we all have this problem.

This is still there. As [00:06:00] we all move to containers, microservices, now like WebAssembly, serverless. This problem is only getting worse. And we felt like giving our experience and our kind of view of this. We had a good chance of making a difference and building this tool that. The three of us who wanted to have 10 years past.

And that was the genesis of the idea. Of course, over time it has, as we. As we talk to more customers, as the product has evolved, we've changed things, but the core vision of developer experience automation and a fast inner loop, and just making it easy for developers to do their thing and not have to worry about all these preparatory steps is core to our company, and it's something that I believe very strongly on, about.

So maybe for those of us that aren't developers on a day to day basis, like specifically, what does it do? How does it help the developer? Yeah. So one of our, I'm going to go with one of our [00:07:00] typical, customers, which is, it's a company who's building software for the cloud. And imagine that your solution has.

And this is becoming more and more common. 20, 30 microservices plus databases plus data. In the old world, you'd have to run all this by hand. Whether it's locally in your laptop using something like Docker Compose. Or maybe you have access to a cluster where you can run this and you have to deploy it manually.

What Octeta does is it lets the platform team pre configure this for you. They give you access to a dashboard. Or a command line tool, and then you go there, you run a single command, Okteto deploy, and all this provision of infrastructure services, building container images, seeding your databases with data all happens really fast and automatically so that you don't

have to think about it.

You click the button, your environment is a femoral environment is up and running in a couple of minutes. You do your work [00:08:00] with Okteto . You can do things like synchronize your code into this environment. Do remote debugging, hard reload your changes so that you can go really fast. And then when you're done working, you destroy the environment and your work is done.

And the next day or the next feature, you do it again. And that way you don't have to think about all these steps. You don't have to carry like the huh. This cognitive load of everything that I need to do by hand before I get ready to work. And how does that sit alongside, for say, platform teams where they're running pre configured cloud platforms and providing that to, say, product teams?

Just describe how like Okteto then would fit into that stack. Yeah, so it's super complementary and most of our customers are platform teams and what they do is they will take Okteto , they will take the cloud provider, they will take all these other things they have. And we give them the APIs and, the connection points to put it all together.

So that, one of our kind of guiding lights here is you have to meet [00:09:00] developers where they are. You can't force developers to change clouds. You can't force developers to change the tooling they're using to deploy their applications. So Ctero gives platform teams the ability to use Ctero as a base.

then add the integrations to their cloud services, their own application definitions, and then they can package the entire thing on, what I think is a, I hope, is a nice UI, a very easy to use dashboard. And then from there, developers can just use it. And Okteto , we have multiple versions of our product, but our most popular one runs on our customer's cloud.

So it's there sitting right next to all their workloads because for us, one of the very important things is we want this ephemeral dev environments to be as close as possible production. And that means same cloud, same configuration, same kind of tooling. as close as possible, right? It's a really nice way to do it.

You've got that integrates with what the developers have already created, because often in tech communities, they get very [00:10:00] emotional about what they've built very close to it. So that's key. So you're not taking away something that they're invested in, but also then being close to the environment under their control, also within their realm.

That's also an important thing to. So people think they're in control of the destiny of what they're about to release. And then you've got that hyper automation that makes everything consistent. So the service managers at the other end in the product groups are all happy because they know when they hit the release button, it's likely to work as long as the code's right, obviously, but hopefully QA.

process, but it's quite a nice model. You're not steamrolling over everything and throwing the baby out with the bathwater, but you're integrating with a preexisting thing. And then that's, you probably find engineers are more accepting of that approach. It's a nice, easy way to get to that hyper automation point.

Yeah, I know you're a hundred percent right. And you touched on something very important that we've seen over and over again, which is in one side, the emotional aspect of We are developers build things. We have tools that you like that you spend time learning. Yeah, I think something that platforms in [00:11:00] the past have gotten wrong, or even some platform teams is coming with this idea of we're going to replace everything you do with this

new way of doing things.

And, it might be better, but the friction that introduces, most developers are like, Hey, I just want to do my job. I don't want to have to learn this to any new thing. So for anybody introducing a platform into an organization, I think that. It's very important to like present this is what we're trying to do with our product as an extension of what you're already doing.

Like in our case, if you're already doing things like using containers, if you're using something like, open source technologies like Helm, Customize or even Docker Compose, we allow you to bring that into the octet of powered environments because we want to be an extension of what we do, and that makes it a lot easier for them to get on board.

And then as they see the value of the platform, then they might start to modify. Some things they do to make it fit better with whatever the platform team's idea of the new workflow must be like, but it has to be this. [00:12:00] Gradual change where developers see value clearly for this to make sense. So in terms then of the business value it delivers, maybe let's start with developing productivity itself.

So what kind of productivity uplifts have you been measuring and is it like significant? It is actually surprisingly significant. A few of our customers are now doing internal surveys, measurements, Dora, all these things. And they discovered a couple of like very interesting inflection points.

The first one is on onboarding new engineers, like one of our largest customers, LaunchDarkly. One of the things that we're really focused on is, hey, we want to build a good engineering culture. where, it's hard to hire developers, it's hard to hire great developers. And when you hire them, you want them to be effective, you want them to feel good about it.

So for them, they had some very ambitious goals of hey, I want to hire somebody and I want them to be effective very quickly. So the fact that with Octetl, your short circuit, how long it takes to [00:13:00] spin up an environment substantially reduces the onboarding time that we're seeing customers where You know, the developers are now being able to push the production on the first week, rather than the first month.

And this is not only the productivity during onboarding, but this sets up the developer, like their morale and their willingness to push the envelope in a different trajectory, because they understand that they work for an organization that cares about the craft, that wants to make it easier for them to...

Get their job done. And just go fast. That's one. And the other is, and Rob, you're talking about this earlier, is as you give developers access to environments that look a lot more like production, instead of just running things on their laptops, the amount of production only bugs goes down. And also the mean time to resolution goes down because now developers have an [00:14:00] instant environment where they can say, Hey, this broken production.

Okay, hurry up, spin up an environment, run it there. You very quickly have a way to reproduce the issue, fix it, release it. And we've seen this happens both because they have access to the same kind of technologies. They are now are used to like using those tools and also because they can experiment.

Sooner in the cycle, then that means they're not like throwing the code and saying, Hey, I hope it will work in production. Now they're more certain now they're building for this platforms and those two things for me have been very interesting because it's definitely in line with what I want as a developer.

We build this very focused on the fast kind of like feedback loop, but then we started seeing this other benefits and now we're focused a lot more on enabling that as well. But those benefits are huge. So a lot of people would write a business case based on developer productivity, onboarding, the tangibles that you can measure, but the intangibles are there about.

A better product in live less risk of [00:15:00] reputational damage, better end user experience. If you've ever been on the end of a flaky website and you need a service out of it is one of the most frustrating things going, but you're much more likely to return to a service that's dependable, always there with a consistent response time.

So being able to engineer in production like environments means you've got more confidence. It's just better for everyone and better business. And I think a lot of people don't appreciate the power of giving the developers access to this type of capability in a cost effective way, obviously underpinned by cloud and the whole ephemeral nature of it means it's there when I need it and it's gone when it's when I don't.

And that means you can work in a very efficient compute. Way as well. So it's not only, but also sustainable as well, isn't it? So it's greener, it's better, it's faster, more reliable, and you get a better end result. I think we need to talk more about that and argue. It's really nice to hear you say that those are the sorts of things that you're bringing to the fore to help organizations do better.

A couple of the things that Rob touched on there actually are a good bridge into where I was going to take the conversation next, because I'm interested [00:16:00] to know the conversations you're having with tech leadership, particularly in organizations that are more legacy in nature. If you're talking to someone like LaunchDarkly, then obviously they're, like a cloud native tech house in their own right, they're going to get this stuff right away.

But what sort of conversations are you having? Not necessarily about the tool set itself, but actually some of the challenges that we've been talking about and are they receptive to it? Do they get the relationship between some of the productivity aspects and developer emotional aspects that we've been talking about and actually.

The fact that they're going to get better products at the end of it. That has been an integrated topic. I love talking about this because this for me as a founder of Okteto has been one of the most. Surprising things because the answer, like the short answer is yes. They care a lot about it more than I initially thought.

Yeah through my job I get a chance to talk to like very senior tech leadership and not like long ago I was talking to the C T O of a global bank and I was very [00:17:00] surprised to hear how all these concerns about. Fast onboarding, developer velocity, morale, willingness to experiment.

We're top of mind for this kind of like super high level leadership. And I'm talking here about CTOs of organizations with 5, 10, 000 developers, right? And what's interesting, and his point to this was, it's like a very interesting kind of twofold answer. One is, When you have more developers, productivity, like anything you do to increase productivity has an outsized result.

A company with 200 engineers. If you give them back an hour, you get, 200 hours. If you have 10, 000 engineers and you're saving everybody an hour, that's like a massive difference in productivity. And the other for a lot of this leadership was around being able to keep people productive, excited, and, to be able to focus on what they do, because one of the [00:18:00] Big difference I've seen between large companies and small organizations, especially

startups, is, large organizations have better processes, typically more clearly defined roles and responsibilities.

Startups can be messy. We're like, everybody does everything and you're trying to like pitch in larger orgs are more augmented. You have the front end team, the back end team, the API team, the whatever. So in this kind of world, being able to automate everything. That you're not supposed to be an expert on is a huge productivity boost.

Imagine that you have in a bank, there's a team that's building the new website for, for your transactions for your like checking your balance, expecting the front end team to be experts on infrastructure on security on, how to spin up an API. It's a waste of time, but if you automate all of that for them so they can click one button, spin up this environment and then they can work on [00:19:00] the UX of the website, which is their core value, their core strength that makes them happier, more effective and just better at their jobs.

And that's something that I think we all feel it. We all feel this kind of the last 10 years we've been, the shift left movement has added a lot more responsibilities to developers. I think these large orgs feel it even more, and it's a mix of different reasons, but I think one of them is this, is that they need to be more specialized because they're bigger orgs, a lot more at risk.

And that's very interesting how this conversation very quickly went from, as you were saying, the kind of more modern, solver first companies. All the way to traditional enterprises, can name names, but on our customer list they're like global insurance companies, global grocery brands, because they build software and they care about productivity.

But the thing, as you touch on, if you get to a level of 10, 000 developers, the amount of effort it takes to get them all in line and ordered and get a [00:20:00] product out of the end is really. Quite an art form on that scale and what you talk about there is by removing unnecessary interactions and allowing you to automate it there's less touch points in the system and then suddenly things work a lot faster if you've ever been in very large complex orgs and there isn't.

The level of automation you talk about and the amount of interaction, because it absolutely kills the productivity of the teams because they're all overlapping and interacting with each other. You get loads of many to many relationships and it just falls away. Doesn't it? Yeah, it's, is absolutely the linchpin of a very large development organizations.

If they don't do this, then you're in trouble. I think a lot of people got their weekends back, right? Yeah, absolutely. Head in hands on a Sunday at four o'clock in the afternoon going, Why isn't it working? Yes, type conversations. And the developer says, Yeah, it works on my machine. Yeah, it works over there in that environment.

Yeah. That is my mission to eradicate that works for my machine and say, Hey, it works everywhere. I'm interested [00:21:00] in some of the things you touched on there about those legacy organizations rather than new software first. So I think our read on this is a lot of legacy organizations quite rightly are looking at transitioning to more.

Platforms and products style working and maybe trying to get their head around what the platform organization should ideally provide into the product domains and into the product towers to enable and not constrain developer creativity and develop productivity in those towers what advice would you give to an organization that's maybe just starting out on that journey trying to understand what that platform capability and organization should be.

That's great for me. And when I talk to this with orcs, like the first advice is platform teams,

whether it's a small team or a full organization, they need to understand that the developers are their customers. I think we need to move away from the old school model where, [00:22:00] you know, the platform teams or it, whatever you want to call it.

It's there to say no and to put blocks and that was like the old school kind of very defensive and they have to move to this world of they need to be enablers. They need to be accelerators. I have a good friend Ivan Pedraza. He used to work at State Street in London. Now it's for Docker and he has this really cool metaphor that he talks about how, platform teams are like, when you're running a marathon, the people who are there to give you water or the pacers, that is a platform team.

Like our role as platform builders is to enable all these teams to do the best job they can. So I think it's very important to understand. Yeah, it's not to control and govern and, support, but in the old notion of Hey, yeah, I'm going to queue you with a ticket. I love the idea of it's like a pacer or giving water or something along those lines.

It's a really different philosophy. Yeah, and it requires a change because it's an art because also it's true that, platforms need to [00:23:00] govern. Platforms need to draw this, the sandbox and hey, this is what you're allowed to do. But if they do it in a way that is like this supportive manner, it's helpful because no developer wants to be faced with Hey.

Here's a blank page of unlimited choices, but by the way, some of them are wrong. And you're going to it wrong, we're coming for you. Exactly. But you have to figure it out. Nobody wants that. In a supportive way though. In a supportive but painful way. We're coming for you. It's so much better if I give you, here's an offer with everything that is pre approved.

Like I have, I was talking to somebody else the other day and they talked about how, you

the chance for you to waste time on like wrong choices. I wouldn't mind that just generally in life. I think that's copyright, cloud reality is that product. But I think there's a, I think there's a product in that. There's a little alarm goes off and said, you're about to make a bad decision. Dave, [00:24:00] maybe you want to reconsider your choices in life again.

I get . Yeah, you have some water. Oh I can see that as like a pin. AI pin says, Hey, maybe you should drink more water instead of another cup of coffee. . Yeah. , . Little gentle nudges in the right way. Yeah. But I think that is actually something that applies to platforms as well especially like large organizations where, and I get it, standardization matters.

Governance matters. If you're running in an org, you want things to be replicable. So when a new team comes in, they can pick up. And that's what platform teams need to like, play those two roles of building this kind of like platform for the 80 percent but they have to realize that developers will find a way the platform if they don't feel supported, find a way to do what they want to do.

Absolutely. Exactly. Exactly. So I think that for me is a, and whenever I hear platform, he's talking about Oh yeah, no, we're partnering with this team to roll out this new version. So we understand what they need and what are their pain points. That's exactly what we do as well.

We talk to [00:25:00] the platform teams to understand their own pain points and taking that, call it kind of product mentality of Hey, you have to assume that people, you want them to choose your platform, even if it's mandatory, you want them to be like, Oh, this is great. Thank you. It's saving me all this time.

That to me is the, yeah, the goal is to have a platform, but it's also something that is. Especially in like legacy enterprises is something you have to think about and actively, maybe even actively fight the culture that might be around like saying no, rather than enabling. But

once you make that shift, and I've seen it done very successfully in like large organizations, like that partnership just flourishes because of course I don't want to deploy databases and have to figure out 35 parameters.

I just want to do my job. So if you give me a big button to say, give me database. Perfect. And he has the right labels and the right size and he's running the right data center and with the right firewall. And I don't have to think about it. That for me is a net win. I'm not going to fight you for let me [00:26:00] configure firewalls by hand.

No, thanks. I want to build, whatever my team's building, not this other layer. So that to me is where platform teams can really accelerate and. Legacy works as they invest in platform teams. This can be a huge kind of accelerator, especially as they move, as you modernize.

So if we talk about then that as the, that's all the benefit of moving to more modern organizational structures, more correctly purposed platform teams and the tool set. I guess by way of maybe just bringing our conversation today to a bit of a close, let's talk about the elephant in the room, the thing that everybody's talking about, which is the.

Impact of AI on all of this. And of course, there's a lot of discourse at the moment about the developer and AI and how that's going to get leveraged. It'd be great to hear your take on it, Ramiro, right at the coalface of this conversation. Yeah, that's a topic you can avoid. It's everywhere.

I was at DockerCon yesterday, two days past in Los Angeles, and one day [00:27:00] was entirely devoted to AI. It's something that is on everybody's mind. My take on that, and it's very similar to platforms. I think that what I see as a feature of AI is In accelerating developers, like even today, you can see things like Copilot or Chad G B T, that you can now leverage them to help you automate those very repetitive tasks that don't require a lot of innovation.

Like the other day I was trying to run a SQL report and I forgot sql. I don't run every day. So I just went to cha say, Hey Chachi, I have this table with this shape. Please gimme a query to extract the data. And it gave me the query I needed it. I could have spent three hours going through tutorials and that would have got it done.

But that saved me three hours. And that, to me, is very powerful. I think that three would feel like, oh, developers are going to be, we don't need developers anymore. I don't agree with that. I think that actually it's going to help developers focus on more high level tasks. [00:28:00] And less on the more routinary, kind of repetitive tasks that we all need to do.

Like scaffolding projects. And what do you think that then? Enables. So if a developer can, with the help of your product, with the help of a I can then really streamline what they're doing. What do you see is the future of the developer and what they can possibly deliver when they're being enabled to that level for me?

One of the things that I see is a developer that is closer. To the problems they're solving and less concern of the shape of the code and like all these things, because at the end of the day, and I think something that, coding is a lot of fun. I'm not going to deny it. I like writing code, but we write code to solve problems, like enterprise developers are writing code to solve business problems and to provide value to customers.

We use code to create that value, but we don't have to, right? We've seen this evolution over the past 30, [00:29:00] 40 years. We're like, we went from this very low level languages, like C assembly to when Java came out, it was like, okay, great. You don't have to worry about garbage collection.

And nobody said oh my God, like without garbage collection, developers are going to be out of jobs. No, great. Nobody likes to manage memory except for C developers, but that's on their own. I think

we're we might not care about code like we do now. And we might care more about putting things together. Maybe that dream of like developers as more like building this complex flows and then letting their AI filling the gaps with code. And then we just write whatever needs to be written.

Almost like a very senior engineer and a junior engineer kind of thing. I can see that as something that AI will allow us to do. And by virtue of that. We're going to be way more effective. We're going to be able to solve a lot more problems. And I find that a lot more interesting, like this kind of high level innovation is a lot more fun than having to write, we've [00:30:00] all written code to write into a database a million times.

And that's not what makes a product win a category or not. It's high level. So that's how I see it.

Sjoukje, what have you been looking at this week? So each week I do some research on related ideas in transformation and tech. And this week I thought we should take a look at How AI can transform the software engineering process. And we already mentioned it, but let's cover it a bit more. So the use of AI powered tools can significantly improve the efficiency of software development processes.

While AI can help software engineers complete repetitive tasks like we already mentioned, it is not expected to replace them. Instead, AI will empower development teams to accelerate app development, write better code, and optimize the software engineering lifecycle. AI power tools will [00:31:00] also speed up the pace of modern software development, promote experimentation and transform the software engineering funnel in the future.

So a question to all of you, we already covered the writing code part of AI where developers can benefit of using AI. Where do we also see AI being adopted in the software development life cycle on short term? You've got the. Let's make developers go fast with AI. But what about the testers and all that sort of stuff?

I don't see much in the testing and security domain yet. Now there's lots of AI in detecting things, but there's not so much AI in accelerating those domains. So you take what we discussed about hyper automation. I think there's still more productivity had down the chain. And what we'll find is. Suddenly the developers will become hyperproductive and then it'll get stuck somewhere else in the system.

So I think there's a sequence of, Oh, we need AI for this and AI for that. And then you'll need a model for the testers and a model for the developers. Cause you can't use the same model for both because you need a bit of check and balance to make [00:32:00] sure that you've not biased things. So I think there's a, there's other domains, particularly in the software development life cycle that quite soon are probably going to romp over the horizon and say, Oh, we need a bit of that as well.

Thanks. One thing I'm seeing that I like a lot is for analyzing production traffic and finding issues. Datadog released a beta at their conference of you have like terabytes of logs and hey, find what's wrong, but models can help us kind of surface. Hey, this API has been returning the right, the wrong result for the last five days.

And you may want to look into this degradation here. I think that everything, anything that relates to like analyzing vast amount of data. It's something that AI will very quickly emerge

to help us just help us get to the hotspots quicker. I don't think it's going to replace someone analyzing and understanding, hey, this is a failure or expected, but finding the hotspots in your logs, in your failures, in your security vulnerabilities, and help you identify which ones you have to quickly fix is something that I [00:33:00] hope AI fixes because it's something that can be really hard, especially on large distributed services.

I also had the view of AI spotting when developers are just writing dodgy code. And they go, spotted Dave's just put a line in the system that looks like it's just not going to work very well in production. Best go have a chat and make sure he doesn't see it. You wouldn't need a very sophisticated AI to spot that, Rob.

Just the idea of the AI is if it just deletes your code after you've written a line and said, try better, try harder. Sorry. Sorry. Try again. You could do better next time. I suggest you meet my chat bot friend who might be able to give you some training.

Why don't you go outside and look at the pretty bunny rabbits and let the AI do your job for you? What do you think the timeframes are Amira, when you look at the bringing together a lot of this tool set that we've just been talking about, both in the main conversation today and just in Schalke's input here, are we three years away from [00:34:00] intense additional levels of automation, AI driven automation?

Is it five years? Is it sooner than that? What's your thinking? Seeing what has happened over the past 12 months, it's just insane. Like I was talking to somebody brought up this other day, which in Twitter, it was like, or x. com, I guess now that said, Hey, Dali and this imagination that was 12 months ago that they came up.

Yeah. And since then, it's just, it's wild. And to me, it's amazing. Not only how fast is moving forward, how fast are companies building? Useful things on top of it and just like releasing them. That seems to be the breakthrough to me. It's like there is something about what's happened post gen AI that's just unleashed a level of creativity into the world.

That it's oh my god, where did this come from? It's like machine learning didn't do this, the case in point, I was writing something today and I needed a little icon, so I just typed in the natural language of what I wanted in the icon into creator, and then Astra in a white [00:35:00] background popped back, bang, in the presentation, don't have to send it off to the graphics team and Astra or anything, it's bang, that'll do, thanks, bang, oop, save.

And it was just in that. saves you and you add up all those little savings where you can ask the system to be creative on your behalf. And suddenly you've got this massive compound effect. It's just teaching people to remember to use the tool because sometimes you think I should have just done that over there instead and you fall back into your old ways of working.

So there's a bit of, I think it's going to take a little bit of time for the human to remember. Oh, I can do this differently on that nose. What a thought provoking bit of conversation and a great walk through the world of developer productivity and creativity. Ramiro. So thank you so much for joining us on the show today.

Thank you for having me. It was an amazing conversation. Now we end every episode of this podcast by asking our guests what they're excited about doing next. And that could be got a brilliant restaurant. Or film books that I want to go see at the weekend. Or it could be something going on in your professional life, or it could be both.

So Ramiro, what are you excited about doing [00:36:00] next? Oh, many things, but it's early Friday here. So I'm excited that the week's going to be over and there's a couple of new restaurants that in town that they want to try. I live in San Francisco, so the food scene here

is really good. So there's this Peruvian place we want to try out tonight that I'm looking forward to.

Oh, very nice. Very nice. Rob and I were in San Fran for Google Next recently, and we went to John's Grill, and Tunnel Top, the bar, and Tunnel Top Bar. I love that bar. It's such a funny place. Great. Finding it. You need a degree in it is actually Tunnel Top. It's on top of a tunnel. So actually they've named it very sensibly.

It's actually literally that. It's really funny. Just need to know which tunnel. Yeah, Dave tried to find it. That look of confusion on his face was quite funny. And John's Grill is great. It's one of the oldest, I think it's one of the oldest restaurants in the US. Is that right? Absolutely delicious seasoning on their fries.

Absolutely delicious seasoning on their fries. I don't know what it is, but it's lovely, isn't it? We went back three times, let's be honest. It was [00:37:00] popular. We did indeed. So a huge thanks to our guest this week, Ramiro, thank you so much for being on the show. Thanks to our barbecuing 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 [00:38:00]

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 every day by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of nearly 350,000 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