DR: [00:00] I'm Danielle Royston and this is Telco in 20. There's an MVNO out there that has won just about every UK customer service award. I'm talking about British MVNO giffgaff. In 2020, when the company announced they were going all in on the public cloud, a lot of industry pundits said they had sold their soul to AWS, but did they?

[00:31] giffgaff was struggling with putting out releases to the market. Releases took too long. Half of them failed and had to be rolled back. On top of that, it took months to procure a server when it could take minutes on the public cloud with AWS. So, they jumped on the public cloud train and improved their code to production timeline by 1,000 times.

[00:50] By rejiggering their release process and moving their production environment to AWS, they leveled up their IT capability from a measly 12 releases a year to more than 16,000 releases a year. Holy schnikes.

[01:03] Today, we're talking to the trailblazer who spearheaded giffgaff's strategy, Steve McDonald, Chief Operating and Technology Officer at giffgaff. We're going to pick his brain on why giffgaff took the plunge, how the team made it happen, and what advice he'd give to other telco execs who are thinking about the move to the public cloud. So, let's take 20.

[01:26] Steve McDonald is the Chief Operating and Technology Officer at giffgaff. Hi, Steve. Welcome to Telco in 20.

Steve: [01:32] Hi. Thanks for having me.

DR: [01:33] Yeah. I am super excited to hear about giffgaff and what you guys are doing with AWS. So, to kick us off, in 2020, I read a press release from AWS about how giffgaff was moving everything they could to the public cloud. I was super excited to read about that, and I wanted to hear the story of how you went all in on the public cloud.

Steve: [01:56] I guess there's a bit of a backstory to this-

DR: [01:57] Sure.

Steve: [01:59] ... to how we got there and why.

DR: [02:00] Yeah.

Steve: [02:00] It goes back, probably three or four years. Everyone was playing with the cloud in 2016 and thinking about moving to the cloud. We were, as well, but we were doing it without any real aim in mind. It was just because it was cool, and it was out there, and we had some vague idea that it might be better, cheaper, more scalable, faster.

[02:20] Our biggest problem at the time was, giffgaff was about 10 years old. Over those first 10 years, we'd focused on building a member base and building a business. That meant, features had taken priority over everything else, and we'd accumulated a whole bunch of technical debt.

[02:37] That was fine to start off with, but it had started to get unmanageable levels, and we found ourselves in the position where we were trying to release code and it was really, really hard. I'll give you a picture of what a release looked like.

DR: [02:51] Yeah.

Steve: [02:52] A bunch of engineers, and testers, and operators would get up at about 3:00 AM on release day, and they'd make their way into the office, and they'd start to do the release, which was entirely manual. At about 6:00, they'd have everything out and they'd start to do a bit of testing, and then they'd realize that it hadn't worked. Then they'd roll it back.

[03:13] That happened about 50% of the time that we tried to do a release, we'd have to roll it back because there were bugs in it that, we didn't know why they were there. They'd just come up in production.

DR: [03:23] Those guys love their job.

Steve: [03:24] Yeah. A lot of very tired people on a release day, and no one liked it. It was horrible, and the business was frustrated as well because we had all of this great stuff that we wanted to get out to members, and we couldn't.

[03:37] Of course, the next release happens, and not only have you got the changes from the last two weeks, but the two weeks before that as well, because the last release failed. So, changes mounted up and it became a bigger and bigger problem. We kind of took a step back from AWS for a sec and thought, "What's our problem here? Releasing is our problem. This process is our problem."

[03:58] So, we mapped it all out, and it's about 200 steps we went through on a release day, and we worked out that we were spending somewhere between three and 4,000 engineer days every year just doing the logistics of trying to release our code. Not writing code, not testing it, just trying to deploy it, which is crazy.

DR: [04:16] Yeah. I mean, not even focused on the business stuff, like the innovation and all the-

Steve: [04:20] No.

DR: [04:21] ... business requirements, just the logistics of getting a release out.

Steve: [04:24] Exactly. You think, what you could do with that time. Software engineer time, at least at giffgaff, is precious. We could be doing so much more valuable stuff with it. So, we're very lucky at giffgaff that we have a business full of people who are very open. They're open to listening.

[04:40] We went to everyone and we said, "Look, this is our problem. This is the biggest problem we have." We suggested that we take six months of not doing any product development just to fix this problem of deployment. They gave us three, which I thought was a fair deal. You go into the negotiation after six, we got three.

DR: [05:01] You got half of what you wanted.

Steve: [05:03] Yeah, that's it. In those three months, we took advantage of that and we didn't do any feature work at all. What we set about was to totally change the mindset of the way that we did development.

[05:14] Our aim was to go from manual releases that were done out of hours to automated deployments that were done during office hours when everyone was there to support them.

DR: [05:25] Perfect.

Steve: [05:25] If you think about that for a sec, it's a huge mindset shift. So, as an engineer, whether you're on the testing side or the development side, your role fundamentally changes. You're not just writing code. You're not just testing for bugs in someone else's code. You're responsible for writing a bit of code that's going to go to production, and it has to work, and it has to be reliable, and it has to get to production quickly. That was a massive change for everyone.

DR: [05:52] Was there one big change that you guys made in your process that got not only your people to change, but also impacted the process and actually helped you get to your final destination? Was it one big change or a bunch of little ones?

Steve: [06:05] No, it was thousands of little changes. Every team had their own way of doing it. That all had to change. Every person had the role that they were comfortable with, that had to change. All of our ops team were very used to being in control of the releases. On the one hand, they hated it because they were up at 3:00 AM and having to deploy this buggy code, but on the other hand it's quite comfortable. They knew everything that was going on in the system. So, you've got to change that mindset as well.

DR: [06:30] Loss of job security.

Steve: [06:32] Yeah. It is just thousands of little changes. I mean, obviously a big one was a move to fully automated testing. Taking that approach of, if you write a line of code, you're also responsible for testing it.

DR: [06:42] Right, which a lot of time, developers don't like. They want to create the code and throw it over the fence, to someone else to test. I've been part of organizations where we've done that. The code that you write, you're responsible for the testing, and that brings a lot more accountability to the quality of the code that they're putting out there.

Steve: [06:56] It does. Ultimately, it's your code that you're writing. A line of code has no value to us. It's the outcome that you're trying to get with that code that we're looking for. That's the thing that changes the experience for a member, or drives value for the business. So, that's what you have to be accountable for, and that means your code has to work, and it has to be reliable, and it has to be tested.

DR: [07:16] Yeah. You said when you first started, you were putting out releases. About half of them were being rolled out. Give me a scale of how often you were releasing.

Steve: [07:24] Once every two weeks. So, 25, 26 times a year, of which half of those would fail. So, we are probably getting a release a month on average.

DR: [07:32] So, you take the three months, you rejigger the process, you get the mindset change. How long before you started to palpably see the change in, either the number of releases or the quality of the code improving? How long did it take before you really started to see the results?

Steve: [07:48] It took us three and a half months before we started deploying every day.

DR: [07:51] Yeah. Wow.

Steve: [07:53] Yeah.

DR: [07:53] Without the rollbacks? I'm assuming the quality went way up.

Steve: [07:55] The quality went way up. It was much more reliable. It still was not perfect by any stretch of the imagination, but we were deploying in an automated fashion every day. We reveled in that for a little while, maybe two or three months, patted ourselves on the back. Then very quickly, the noise started to come from our engineering teams that, "God, this is so slow." On the one hand you're like, "Hold on. We used to deploy 12 times a year. We're deploying every day now, and now it's slow?"

Then actually, I think that was a really good thing because the discipline in our teams had gone up, and their bar was set much higher, and they were starting to complain that once a day was too slow. So, we opted to twice a day, which wasn't dramatic at all. It was just running the same pipeline twice.

DR: [08:40] Had you moved your build process onto AWS?

Steve: [08:42] No, we're still totally in our data center doing this.

DR: [08:46] Wow. Okay. Got it.

Steve: [08:47] So this is, I guess, probably 2018, 2019, something like that.

DR: [08:52] Yeah.

Steve: [08:52] It's at this point as we started to go, "Well, what's the next stage? We're doing deployments twice a day. It's still a bit slow. It's still a bit clunky. We've got this monolith, so we're smashing everyone's changes together, and that gets complicated."

[09:05] It's at that point that we started to re-architect the system. We initially started to break things off into microservices, to build everything new, as a new microservice and to, again do that in our data center. We started then, to push up against the limitations of our data center.

[09:24] If, obviously you break stuff out into microservices, perhaps it's a little bit harder to run, you need more servers. Deploying a new server into our data center was, charitably, four weeks, often took a lot longer than that. So, we started really finding those limitations. We also had a massive database outage as well. A couple of controller cards in one of our servers went, and it left us without a database for about six hours, which was pretty catastrophic. That really prompted us to go, "Right. How can we accelerate, now, our journey to AWS?"

DR: [09:58] Yeah.

Steve: [09:58] "What we're after is speed of deployment and reliability." Those were the two reasons that we were moving to AWS. So, it felt at this point, we started to coalesce behind the reason for AWS. It wasn't just cool and vague. It was, "No, we want speed and we want reliability."

DR: [10:16] Purpose.

Steve: [10:17] Yeah.

DR: [10:17] So, you're in a data center. Did you guys lift and shift and then start to retool with AWS services and products? How did you guys lift what you guys were doing and get it to AWS?

Steve: [10:30] We deliberately didn't do a lift and shift, and partially that was because we felt like we were going to take our existing systems with all the compromises and flaws that they had-

DR: [10:41] And put all your problems in AWS?

Steve: [10:43] Yeah. We weren't sure that was really going to get us the benefits that we were looking for. Would it be any more reliable? Maybe a little bit, but probably not an awful lot. Would it be any faster? Well, no because our build process was a couple of hours for that build pipeline to run. So, we really wanted to break this down into independently deployable services. We started to build everything new on AWS in our target architecture and decided that we would come back a little bit later to the big, nasty monolith, and we'd move that a bit later.

[11:15] That started to work quite well. This was when we really started to see a bit of an explosion in the number of deployments that we were doing. So, whilst our legacy stack was still going twice a day, I think over the course of about 12 months, moving into AWS, we went up to about 6,000 production deployments a year. So, real step change. Partly, that was down to AWS, and being able to stand up infrastructure quickly, and all of that good stuff, but partly it was down to the architectural change as well. We were starting to deploy independently rather than deploy one big lump at a time. I guess, probably the next thing that happened was, we got a bill from AWS.

DR: [12:02] It was big?

Steve: [12:03] Well ...

DR: [12:04] Or bigger than you thought.

Steve: [12:05] It's a pay-as-you-go model. You pay for what you use. So, it's entirely fair, but I think we had, maybe undercooked how much we thought we were going to use.

DR: [12:14] Yeah. Yeah.

Steve: [12:15] We took a close look at it and we realized that, actually a big part of this cloud strategy for us was going to have to be cost optimization. We're quite proudly a very lean business, and we needed to make sure that discipline of being efficient, we took over to the cloud as well.

DR: [12:33] Yeah. I think this is a common thing when people force move to the cloud, they do a case study on the back of an envelope, or a very detailed Excel spreadsheet, and then they move in, and everyone's really excited. It's so easy to spin up a side server or a little test thing, or, "I'm going to go play with something," and you start to see those bills. There is a little bit of bill shock that happens. So, I always tell people, that cost optimization team, people thinking about what you're using and how you're using it, how you're managing and tagging servers and things like that has to start pretty early. You can't move a bunch of stuff and then do it. It's a little bit harder.

[13:13] It sounds like you guys started to build up your cost optimization chops, because it's not just financial, it's also a little bit technical because there's an element of, "Well, what's the workload? What tools are we using? Should we use this database over that database?"

Steve: [13:29] Yeah.

DR: [13:29] It sounds like you guys, pretty early in the journey were like, "Whoa, we need to start to pay attention to what we're using and how we're using it."

Steve: [13:35] Well, there's the easy stuff, the discipline of, "Don't leave stuff hanging around if you're not using it." That's very easy to cover off, but we ended up re-architecting our solution again, to optimize for costs. So, we run our entire production estate on EC2 spot instances now, and they're the excess compute that AWS has, that is currently not selling to anyone, and then they sell it off very cheaply, and that's brilliant. The flip side of it is if someone wants one of those, and we'll pay full price for it, they'll take it away from you with, I think it's 120 seconds warning.

DR: [14:07] Two minutes.

Steve: [14:08] Yeah. You have to be in a position-

DR: [14:10] To move your workload pretty quickly.

Steve: [14:12] ... where within two minutes you can build a new server, you can insert it into the cluster. We do that, now about 60, 80 times a day, build a new server, stuff it into the cluster because one of our spot instances has gone away. It's things like that, you know?

DR: [14:27] Yeah, that's pretty high capability on the cloud journey, at least from telcos. That is absolutely a great pattern to use, and it means that you have a high capability on your team, to think in that way, take that risk. Understand that a server might be blown away with two minutes warning, and then be able to rebuild the server quickly and still keep running. I think that's amazing.

Steve: [14:46] Also, you're making sure that your infrastructure as code is actually representative of what's in your environment. Quite often, we see drift there, where someone makes a change manually to something and doesn't reflect it in the code, and all of a sudden those two things aren't aligned anymore. Rebuilding stuff frequently makes that a non-problem, which is great.

DR: [15:09] Where are you today? You said you'd taken up releases up to about 6,000 releases a year, up from 26. Then today, where are you guys? How often are you putting out, I guess, now they're microreleases, but how often are you pushing code to production?

Steve: [15:22] The last couple of years, between 15 and 16,000 releases a year, and that seems to be pretty stable. That gets us the efficiencies around deployment that we've been looking for, and we can focus on other things as well.

DR: [15:35] Yeah. If you had to do it all over again with all your lessons learned, would you go back in time and tell 2017 Steve to do anything differently, or, "You got to go through this journey. This is a pretty typical journey"? What advice would you give to another telco exec who's embarking on an all-in move to the public cloud?

Steve: [15:53] I think you do have to go through the journey. I think it's not about the cloud. The cloud is a part of it. Having a solid reason for it is really important. Making the time to build your architecture in a way that it can take advantage of the cloud is really important. I think one of the biggest things for us, what made this a success was the support that we got from the rest of giffgaff.

DR: [16:15] Yeah.

Steve: [16:15] They listened, they understood the problems, they felt the pain, and they placed trust in us to fix it. I think sometimes that's quite rare to find.

DR: [16:23] What I love about your story is how much you're bringing. I call it HR change management. Change management really is an HR strategic move. People sometimes think of HR as just benefits, and onboarding, and things like that, but so much of it is a receptive culture to this kind of change. You highlighted that the business was willing to pause on all features to let you figure it out. There was probably experimentation and setbacks, sometimes if you're not in a supportive culture where people understand that you're learning, and there's going to be some failure here and there, but there's a long-term goal.

[17:00] So, it sounds so much like, not only your own team, the IT people and the developers understanding what you're trying to do, but also the larger organization being supportive. I think that is so critical to big changes, like what you guys did, which is completely revamp your release process. Move it from under your own roof and your own data centers into the public cloud. That is so key, is having great support, culturally from the people. That sounds like that was a big part of your success.

Steve: [17:30] Yeah, absolutely.

DR: [17:31] Awesome. Well, speaking of culture, I hear you are a classical pianist with the team. Who's your favorite composer? Is it Rachmaninoff, Mozart? Is there a particular composer's music that you love to play? I think you're getting your diploma in classical piano.

Steve: [17:49] I love playing a bunch of the classical and romantic composers, but Chopin's probably my favorite. Rachmaninoff is a close second, Beethoven is a third, but there's so much to choose from. The repertoire in piano music is absolutely enormous. I'm slightly in awe of all of these composers that lived hundreds of years ago and did more in their short lives than, I think I could ever hope to achieve in mine.

DR: [18:14] No. That's amazing. My mom forced me to play piano. I played tennis, and the rule was, if I was going to play a sport, I also had to play the piano. In my state, I live in Texas, there's a famous piano competition. It's held every four years, called the Van Cliburn International Piano Competition.

Steve: [18:33] Oh, he's a wonderful pianist.

DR: [18:34] They most recently held it in 2022. Steve, if you're ever in Texas, I guess the next one's in 2026, maybe we can go check it out.

Steve: [18:43] Absolutely. I'd love to.

DR: [18:44] Yeah. Awesome. Well, see, this was a great discussion about what you guys have done with AWS and your release process. In a short period of time, you guys have completely changed it, and wish you guys the best of luck. giffgaff always wins best customer experience, and so continued success to you guys over at giffgaff. Thank you so much.

Steve: [19:04] Thank you.

DR: [19:05] Stick around because we're ending each podcast with a Telco in 20 takeaway. I have 20 seconds to tell you something you need to know. Listen to what Steve said in our discussion. He said he couldn't have done it without the support of the company at giffgaff. When you're driving big organizational changes like Steve, a lot of people think it's just about the tech. Really, as a telco executive, you will need a strong HR partner that can help you map out the cultural changes required to complete such a revamp.

[19:35] You need to think about the processes that will change, how job responsibilities will be adjusted, and obviously the mindset shift that will need to happen. All of that is super critical to driving a successful cloud transformation.

[19:48] My mentor, Jim Abolt, has a great framework for driving change. He always likes to say, "A great idea is nothing without great execution." Take Steve's idea about revamping the release process at giffgaff. It wasn't about the technology. It was about getting the whole company to understand the end goal. Being supportive through the change and driving it to the finish line. Super impressive.

[20:10] You know what else is super impressive? My conversation with Jim Abolt about change in episode one. It was our first episode, so it might be a tad cringey or awesome. Follow us on Apple Podcast and Spotify, and leave us a review. Connect with me on Twitter @TelcoDR and on LinkedIn. Check out our awesome YouTube channel and sign up for our killer email newsletter on telcodr.com. Later, nerds.