CHAEL: I did actually notice that you YouTube Advent of Code. If you ever want help with your overlays, use, like, the box in the corner. You can put, like, a little box around that box, and it'll look real nice. And you can just move it. And you can, like, put chat in there, maybe? Pretty overlays.

JARED: That's a good idea. That's a good idea.

[laughter].

JARED: Welcome to Dead Code. My name is Jared Norman, and today, ChaelCodes is critiquing my stream overlays. No, that's not what we're actually talking about. Chael is on here to talk about the benefits of streaming and other programming activities done in public, contributing to open source, and why you should not just consider doing that, maybe outside of your work, but how you can push for it within the work that you do. Let's get into it.

Hey, Chael, welcome to the podcast.

CHAEL: Hi, Jared. Great to be here. I'm really excited to be on the Dead Code Podcast. I love the music that you have for the intro and the outro.

JARED: Awesome. Thanks so much. I was really happy with how that turned out.

For our audience that aren't familiar with you and haven't seen your streaming and everything that you do, could you introduce yourself to them?

CHAEL: Sure thing. So, my name is Chael Wright-Munn. I go by ChaelCodes online. I'm a content creator, so I'm on YouTube; I'm on Twitch, been on Ruby Social. I've recently been on a little bit of a hiatus or a maternity leave, and I'm excited to get back to some content creation. I'm mostly known for my live streams on Twitch, where I play programming games and do open-source streams. I added, live on stream, the comparison validator to Rails, added Twitch clips to Dev2, and recently I've been doing some collaborations with GitLab.

JARED: Very cool. Very cool. So, in a previous episode, in the episode with Stephanie, your talk from the RubyConf in San Diego came up, where you were talking about sort of the benefits of streaming and putting yourself out there. What's the idea behind that talk?

CHAEL: Oh my gosh. Okay. So, I love that talk. The core idea of that talk is just kind of an explanation of how live streaming has changed my life and how it can change the lives of other people as well. But the really big part of it, for me at least, is doing something outside of your company and really building a name for yourself, a reputation for yourself, and a library of examples. This is something that I've talked about in a couple of my talks, where I'm just like, I think you need to have a library of examples that exists outside your company that shows the code that you've created.

The way that I've done this and I've built this is through my live streams. Like, my live streams are why I did the things that I described earlier. The content wheels are always churning, and there's always something new that you need to create in order to do a live stream. Like, you have to be writing some code. And so, what I've seen is I've seen my growth just skyrocket. My capabilities as a developer increase rapidly because I'm talking to all of these other developers out there that are coming by my streams.

And it's just an incredible experience to learn from these other people, teach what I've learned to other developers, because you're getting people from, like, all ends of the experience level. And I think that it's an incredible way to put yourself out there into the community. And so, I want to encourage other people to livestream, but I also want to encourage other people to find their own way, to find their voice in the community, and to put themselves out there, even if it doesn't necessarily mean live streaming.

JARED: So, you describe yourself as a content creator. Is this something that applies to us regular programmers who aren't necessarily interested in getting into, you know, being content creators specifically?

CHAEL: I think so, for sure. I think that if you've ever sat there and written a blog post, even if it's for yourself, even if it's for your colleagues, like, you're, in a way, kind of creating content. And I think that when you put those blog posts out there, you're creating notes for yourself to reference in the future, right?

So, when I think about content creation, I think about storytelling. I think about these examples. And those are things that benefit you regardless of whether or not you're trying to build an audience. So, like, when I think about content creation, I'm thinking about creating artifacts, art, blog posts, like, the things that exist as content to be consumed. But I'm not necessarily thinking about the impact it has on an audience.

And I think that when you talk about a regular developer versus somebody who's more public-facing, that's the real difference. It's like whether you're playing to an audience. But I think that we are all content creators. Like, every time we make a PR, in some ways, that is content. And I think that improving those skills can improve the code that you write, its readability and also the PR descriptions that you write.

I just...this is something that I'm really passionate about, and I'm trying to give you space to ask questions. But when I think about content, I think about trying to take something and package it for other people to consume. And, in a lot of ways, that's what we do with a PR is we take our code, and we package it with screenshots of the changes that we've made, a description, an issue to explain the problem that we're trying to solve, right? Like, we are creating a piece of content with every PR for others to consume.

JARED: Yeah, and, you know, I definitely agree. I create, you know, a fair number of PRs. And the reality is those only get seen by a very small number of people when I'm, you know, when

I'm working on the projects that I work on for work. So, you're saying that I should be looking for opportunities to take that energy and take those communication skills and sort of broaden them out to a wider audience.

CHAEL: Oh, absolutely. And, in particular, like, when I think about this, I think about open source. Because I'm working on Twitch, my code is out in the open, so everything I do has to be out in the open. And I think that when you're releasing things out to the open-source community, first of all, you can access it later, right? Like, you can see it, and you can use it.

My big example for this one is, early on, I had an issue where I was using Windows Subsystem for Linux, and I'm streaming it. And, like, the problem I'm having is Docker with Windows Subsystem for Linux causes this issue with the file checker in Rails, right? And what was happening was I would have to restart my server every time I wanted to update my files.

Now, I was working on one of my open-source projects when I did this, right? And if you change the file checker to an older version, then it will actually reload without you having to reload the server. And I was actually rewatching one of my videos recently where I was working on Dev2, and I ran into the same problem. And what I did is I went and I searched that repo, and I pulled it out, and I literally grabbed that moment.

And the thing is, I've had this happen so many times. Like, when you change jobs, and this happens, like, two to three years in our careers in tech, and it feels like every single example that you've created, all of the learnings that you've had, all of that kind of gets wiped out every time you swap careers. And if you're building it out in the open, if you're building it for yourself or for others, then you don't lose that knowledge. You still have the code to reference. You still have the PR that you created. You don't lose all of the things and knowledge that you've accumulated. You know exactly where to find it.

JARED: Yeah, that’s almost sort of a parallel. One piece of advice I give people when they're leaving a company, whether it's, you know, leaving my company or they work somewhere else, is I remind them not to remove their work email address from their GitHub account because it will untether them from all of your contributions. So, if you have this nice, green graph sitting there and you remove your work email...because it'll show that there was activity in private repositories. It won’t show what it is. If you remove your work email, it disconnects you from all of that activity and, like, all of the commits and everything.

CHAEL: Wow.

JARED: And you'll lose all of that commit graph. And as long as you...you can just leave those old work emails on there, as long as you don't have any, you know, emails set up to go or, like, any of the settings, so nothing should be going to those email addresses. But this is taking it a whole ‘nother level. This is, you know, saying that's private activity. No one should really be judging you based on your commit graph, your activity graph on GitHub, but let's try and move even more into the public sphere and to lose even less as we move from job to job.

CHAEL: It's the same concept of lost history, where it's like, you have this history, this reputation that you've built up within your company, and if you stay within your company, that's great. You have access to all of that history. You have access to your green commit graph. The moment you swap companies...and, like, we're in tech; it happens so often, and oftentimes, it has nothing to do with the developer in question. You have things like everything from layoffs to changing company cultures. Like, a lot of times, you don't have a choice in whether or not you're leaving behind your history and your reputation.

JARED: Yeah, something else I've kind of thought about because my company works on an open-source e-commerce platform, and we do some contributions back to the platform where we can. But the bulk of our work is on the actual stores that use the platform, and that's all proprietary. That's all closed source. And we try to find opportunities for those different stores to collaborate and build open-source extensions to the platform.

And there are a variety of benefits to that for those individual stores. They're sharing work. They're sharing the workload and potentially getting the benefits of open-source maintainership, that kind of thing. Other stores may choose to use it and contribute back and improve those extensions. But I've never really thought about it from the context of, when we do that, as individuals, we should be incentivized to look for those opportunities as well because those will leave behind this kind of artifacts that you're talking about. And it sort of makes me view the work that we do in sort of a new light and see, you know, more value in it.

CHAEL: I'm loving this concept of, like, artifacts and this way of describing it and, like, describing it in terms of history because that really makes it obvious that it's broader than just, like, content creation, or Twitch, or YouTube, or whatever specific platform that you're making an impact on, right? Like, you want your code to go further than your company. You're looking for an impact radius that is larger than that. That's really what we're talking about here and finding a way to do that.

JARED: Yeah. So, you know, beyond, you know, we've talked about streaming and open-source contributions. What are other ways we can approach the work we're doing and find new ways to carry it forward and not lose these artifacts?

CHAEL: Oooh, are we going outside of ourselves? Because if so, I almost think, like, we should be talking about mentorship as well because we're talking about the impact that we have on other people and on growing them and changing their experiences. You know, it's kind of interesting because you mentioned working between these different companies in order to contribute to open source and how they can kind of, like, feedback into each other.

You talked about how, like, if you're working with one company and they work on that e-commerce platform and contribute to it, all the other companies can use it, but then they can also use what the other companies produce. In that same way, like, mentorship and helping people, like, you have access to their examples and their blog posts. Like, the impact that you

have on other people is just as important as the impact that you have on a specific codebase or in a code way. It's kind of like with open source how the hardest parts of it are not actually the code in any way shape, or form. Like, the hardest parts of it are the people.

JARED: Yeah, no, no, absolutely. And that's something that, you know, those broader systems and sort of fixing them was part of why there was a big split in the open-source community that...I'm a part of the Solidus community from the previous project, Spree, was that the various factors of the management of Spree were creating sort of splits in the community, where it's actually very appropriate here.

Because upgrades were becoming very, very difficult, everyone is being stuck on forks of arbitrary versions, maintaining their own customizations, and backporting security fixes. And it meant that when anybody wanted to make changes to the platform, they were not incentivized to get those changes merged into the upstream open-source project because they weren't on the latest version. They would never see those changes. They needed them on their private forks.

And this created a tremendous amount of waste in terms of people duplicating work, but it also meant that all of that work was hidden. Whereas when we fork the project, part of what we want to do was make the project really easy to upgrade so that everyone could stay up to date. And if people needed to or wanted to contribute changes to the platform that would benefit everyone, they would be incentivized to do so.

And that has had a sort of side effect of exactly what you're talking about, more people working in the open, making more contributions to the open-source side of things. So, it's kind of interesting that this sort of interesting management change on how the project was run has sort of trickled down into, you know, helping people surface their work and their careers and incentivizing them to do so and making a space for that.

CHAEL: Definitely. And I like the idea of thinking about it in terms of priorities, too, right? Because you've decided to prioritize making it easy to upgrade because you value those open-source contributions.

JARED: Yeah, I think the, you know, the systems that surround open-source projects, there's a lot of different ways to organize those projects and incentivizing contribution back is one that reaps a lot of benefits. There's different reasons to run open-source projects, but that one's very important for their long-term health and certainly makes them more appealing projects for you and I to contribute to.

CHAEL: No, absolutely. When we were talking about this, and you're talking about the upgrade path and, like, making it easier to upgrade, it actually reminded me of a talk by Eileen about upgrading Rails inside GitHub because they were on that same fork. And now they've managed to get GitHub on that master branch, so they're able to contribute directly to it. And they're able to very heavily and strongly contribute to the success of Rails and its improvements.

JARED: Absolutely. Do you think developers at tech companies should be pushing for their teams to keep things really up to date so that they have more opportunities to potentially contribute back to the open-source side of the tools that they're using?

CHAEL: 100%. And, like, I am a huge proponent of open source. In fact, at my last company, one of the things that I did was I created a proposal that would allow developers to use open source on their e-time. E-time is basically, like, employee time to work on technical debt or anything like that at this company, and it was 20% time. And I made the argument that open source was as valuable as dealing with that technical debt inside the company and that contributing to open source could both improve developer skills, improve their ability to reach out to the community, to reach out to the people who are maintaining these projects, and also, to just, in general, improve their skills in a way that was as valuable as dealing with technical debt.

JARED: I think that's probably something that developers should be considering when they're looking at how they spend, you know, development time, or e-time, or whatever they have at their company is some potential avenues for use of that time is going to produce these artifacts that are going to live on beyond their time at that company.

CHAEL: Absolutely. And, you know, when we're talking about developers at their companies, one of the big things when you're dealing with a problem is deciding where you want to solve it. So, for us, we ran into a problem at one point where a gem that we were using, it was like an SDK gem, so, mostly, it just called all of the APIs that this company added, was missing one of the APIs.

Now, we could have solved that on our end by just making an API call out having, like, a whatever, a HTTP request somewhere in our application. But instead, what we did is we took a little bit of time to go to that gem, make that change. And because we did that, we were able to kind of get in touch with the maintainers of that gem. We were able to fix it for anybody else that needed to make that call.

And I think a lot of times what I see is people that kind of, like, monkey patch things inside their company and inside their systems because they find a problem, but they don't even contribute issues upstream. Because one of the things that can be really valuable for open-source repositories is opening issues to let the maintainers know there's an issue and to let people who are interested in investing and contributing outside of their work hours to go work on that issue and solve it in a way that could affect everybody who uses that gem.

JARED: Yeah. And that sort of touches on, I think, a benefit we've maybe not talked as much about. We've been talking about, you know, the artifacts that these activities produce. But it sounds like through that activity, you also were sort of building your network, and your company's network, and their relationship to different people and open-source projects and the people that work on them. Is that also a sort of a significant part of working in the open like this?

CHAEL: Oh, absolutely. Like I mentioned, I've been doing the collab with GitLab. If I didn't stream, would never have, like, talked to or met Lee Tickett from their collaboration team. I was recently re-watching the video where I added Twitch clips to Dev2, and from that, I ended up meeting Nickytonline. And we tried to fix, like, [inaudible 18:40] Docker setup after that because I'd experienced a lot of issues with it. Wasn't exactly successful with that one. But when I think about it, I personally am making these connections and meeting people. Like, our conversation right now is happening because I live-streamed, right? Like I'm getting to meet you. Hello. Nice to meet you, Jared [laughter].

JARED: Good to meet you [laughs].

CHAEL: Exactly. Like, when I think about, like, consequential friendships as well, like, Matthew Draper is one of my best friends. And he showed up in, like, what is it, some of my earlier live streams. He's actually the one that told me about the file checker fix there. And he's been an incredible mentor, and he actually recommended the, like, talk, like, the subject for that lightning talk.

And when I think about, like, all of the people that I've met, like, what is it? Corey Haines used to come by stream, and he would, like, watch the stream while he was practicing piano. So, like, I got to learn all about how he would do, like, code retreats. And, like, my entire position...so I worked as a DevRel at New Relic. I would never have had that opportunity without streaming, the opportunities from working in the open.

And that's really, like, you know, the core of my talk was like, oh, you should live stream. And it kills me, but, like, the last two slides were me going, hey, it doesn't have to be live streaming. You just have to work in the open in whatever way works best for you. That's really the key to it is, when you work in the open, you get to meet incredible people, and you get to expand your network and meet people that you wouldn't have otherwise.

When I was in college, one of my friends said that I was a big fish in a small pond, right? And that always stuck with me because I was like, I don't want to be a big fish in a small pond. Even if I'm the smallest fish in the ocean, I want to be able to swim free and get to meet all of the other brilliant minds that are out there.

JARED: Yeah, agreed, agreed. So, let's say I want to get started with streaming hypothetically, not me, actually, because I have streamed before [chuckles].

CHAEL: Advent of Code.

JARED: Yeah, yeah, exactly. I'm thinking of streaming Advent of Code again this year actually. I might actually do it. Looking at my schedule, I think I have time this December to make it happen. But for, you know, our hypothetical audience members, how do you get started with this stuff?

CHAEL: Okay. So, the first thing it kind of breaks into two parts, technical and community building, right? So, from the technical side of things, a lot of people start with Streamlabs OBS as, like, their tool that they use. Some people use...I think it's called StreamYard, which, like, allows you to basically kind of do everything online, and it's like a SaaS platform for streaming. I don't really like that one because it doesn't support captions the last time I looked.

And also, if you start with StreamYard, you're kind of locked in there, and you're locked into their monthly fees, and you don't have the opportunity to, like, learn to work with actual desktop applications. So OBS, Open Broadcast Software, is what Twitch Live and Streamlabs OBS, that I just mentioned, and StreamElements are all based on. So, that's an open-source broadcast software. And what it does is it allows you to build scenes, to actually, like, build out your stream.

I strongly recommend that people get a decent camera and a decent mic. It doesn't have to be, like, top-of-the-line amazing. It has to be decent, with good reviews. The reason I say decent is because there's a lot that you can actually do to your space to improve baseline gear, right? So, for example, if you have a MacBook with, like, a regular webcam, you can dramatically change what that webcam looks like by changing the lighting in the room.

So, for example, I have two-point lighting setup, and that makes, like, my face look really crisp and clear, even if I'm just using a MacBook, versus if you take the MacBook, turn it around, like, you're backlit or anything like that, you're going look awful. So, like, it doesn't matter how nice your camera is, like, it kind of matters, but, like, there's a baseline quality level, and then, beyond that, you're kind of fine.

I'm also assuming that we're talking about coding live streams as well. As far as microphones, like, noise dampening and making sure that your room is quiet, making sure the air conditioning is off, making sure that there's no electrical interference right next to the microphone. All of these things can make a condenser microphone sound amazing, even if it's not, like, top-of-the-line quality. So, those are the two things from, like, a technical perspective that I think you really need, like, from an AV perspective.

I will say that the microphone is significantly more important than the camera. A lot of people go with, like, no face cam. Like, one of my favorite streamers, Uncle Scientist, he doesn't use a face cam. Another of my friends who's an artist, QuinnStone, she doesn't use face cam. But I do think that if you're streaming, it's important for people to be able to relate to you and connect to you, and I think camera helps with that. Like, right now, as we're recording this podcast, we can see each other's faces, right? I think it helps to develop a connection, right?

The last thing I'm going to say is audio is much more important than video. The reason being a lot of people listen to streams in the background. They'll listen to it while they work. So, they're hearing your audio, but they're not necessarily paying attention to the video or the code. Some people do. Some people absolutely are, like, reading the code along with you, but some people just kind of, like, pop in and out and, like, use it as background noise.

Okay, now that we've covered the technical stuff, that's the easy part, right? Is like, get your software to stream, get your camera, get your mic. The hard part is actually the community building because a lot of people, like, they start streaming, and there's nobody there. And they're like, I don't understand. I have the best camera. I have the best microphone. I have an amazing stream with, like, beautiful alerts and, like, custom this or that, but there's no one here.

And the important thing when it comes to building community is to remember that you are building friendships and relationships with the people in your chat, right? And, like, you don't show up to somebody and be like, "Well, I'm the best and the shiniest and the nicest and the most wonderful person. So, obviously, you are my best friend." That's not how it works. Like, you develop relationships over time. People become invested in you as a content creator over time. So, I think a lot of people try and just create an amazing stream and hope for everybody to show up, and it doesn't happen. And then, they get discouraged, and they quit without taking the time to invest in it and see where it goes.

The other big thing when it comes to community building is discoverability. Twitch is terrible for discoverability because you're basically, like, here is an image of my face and one thumbnail. Would you like to spend four hours with me? And, by the way, you're going to have to watch a 30 to a-minute-long ad in order to do that. That's really high investment.

JARED: Yeah, that's tough.

CHAEL: [laughs] That's a tough value proposition. So, what you want to do is you want to find other platforms, Mastodon, I love Ruby Social, YouTube Shorts, TikTok. Twitch has started to implement shorts as well, and I'm not sure how successful that is. But it does help with discoverability. And just kind of think about other ways to get yourself out there. Talks are a great example of a way to get yourself out there and, like, communicate with the community. And if you've got a YouTube video afterwards, that can be another type of artifact.

JARED: Right, yeah, that's a good one.

CHAEL: I went through a lot of stuff right there. I should make a blog post.

JARED: [laughs] Yeah. Yeah. Yeah, some kind of artifact.

CHAEL: Yeah, a primer to streaming [laughs].

JARED: Yeah. How do you decide what to code, what to work on your stream?

CHAEL: Oh my gosh. That's, like, a hugely important part of this that I completely left out. Thank you for bringing [chuckles] that up.

For me, what I actually started with was programming games. So, I started with a game called Regex Crossword, where I would solve RegEx crosswords and explain how I'd solve them and what I was working through. This remains some of my, like, most popular YouTube videos, and people will be like, "Oh, I'm in class learning RegEx. Thank you so much for explaining this. I understand it so well now [laughs]."

But the thing I liked about it is it was a structured format, right? So, like, I knew what I was getting into. I knew I would have something to work on. On a couple of different occasions, I've tried kind of like the talk show thing. I tried doing Pomodoro, where it was like, I would focus, and I would spend five minutes talking to people. That really doesn't work for me. I need something to do and focus on and talk about.

So, I kind of break that down into a couple of different things. I break it down into, like, short stories and epics, right? So, I've had two large applications that I've worked on for a long time. The first one was HuntersKeepers, which was an app for managing games of Monster of the Week. I didn't finish this app. My campaign fell apart before I finished it, as most role-playing games do.

The second one [laughs] is one that is still ongoing, which is called MeetAnotherDay. It was previously known as ConfBuddies. But, basically, the idea is if you're going to a conference, like, it can be really nice to know who else is going and which of your friends are going. So, the app is based around, like, you build out your profile. They build out their profile. And then, you have friendships, and you have events, and you can see who's going to the events. It's very permissions-forward.

But that one I would describe as, like, my epic, right? Like, I always have something I can do on it and something to work on, and I know what the next step is. And I've got, like, a long roadmap planned out. So, for, like, the majority of my content it'll be something that's part of the epic, particularly around Hacktoberfest. Hacktoberfest has always been, like, a really big thing for me, where I've tried to push myself outside of my comfort zone. I know it's not popular with a lot of people, but it's been really helpful and good for me.

So, Hacktoberfest, what I've tried to do each time, like I said, push myself outside of my comfort zone. I'll go find four different open-source repositories, and I'll contribute something to them. In my eyes, this is like a short story. So, for example, the Twitch clips that would be a short story because I worked on it one day for 10 hours [laughs], a long short story. But that's, like, a story from beginning to end. That's the other big thing I do with my streams is I try to think about it as stories.

So, like at the beginning, I'll have an issue in mind that I want to work on. And, at the end of every stream, my goal is to have a PR. I've had a handful where we didn't have a PR that stream. It just did not work out [laughs].

JARED: Life could happen.

CHAEL: Yep. The challenges were insurmountable. But that is my goal is to have a complete story beginning to end, issue to PR.

JARED: Yeah, that makes a lot of sense. And, you know, trying to have a project that you can work on sort of ongoing, if there are not other things. I really like the idea of Hacktoberfest being a streaming thing because it shows, you know, you're actually putting the effort in with these contributions you're making.

You know, one of the complaints is people just make drive-by contributions to projects that are potentially not as desired by the maintainers, whereas this, you know, this shows, you know, you actually went to this project, found something to do, put the work in. You weren't just messing around. You were learning about the project. You were spreading knowledge about the project, and you were finally making that contribution to it. So, I love that approach to Hacktoberfest. That's really great.

CHAEL: Absolutely. One of my favorites, I think, was I've contributed to Code Thesaurus by GeekyGirlSarah, who's one of my viewers and she also streams as well. But that was just kind of a really cool moment where the two of us, like, had an opportunity to work on her project. Code Thesaurus, by the way, is, like, if you have some code in Java and you know how it works, you can take concepts in Java and translate them to concepts in Ruby.

JARED: Oh, that's really cool. Like an interactive Rosetta Code kind of experience?

CHAEL: Not so interactive, more documentation-based. But it's really cool and it's very easy to contribute to. And the reason I bring it up is kind of like that idea of Hacktoberfest being an opportunity because I'm not sure I would have contributed to it if I weren't looking for four different open-source repos to work on. And it's a great example of, like, bringing attention to a project, networking with a peer, and also learning something new.

JARED: Yeah, it sounds like there are no end of benefits to not just streaming, but, you know, working in public and collaborating with others, building your network, building a repository of all the kind of work that you can and do in your work as a programmer. So, where can people go to follow you online?

CHAEL: Yeah, okay, so I'm ChaelCodes on most platforms. Currently, what I'm doing is I have chael.code/links, and that's got links to all of the social networks that I'm active on. I think I need to add my Signal app because I don't have a good way to, like, DM, or message people because I'm not on Twitter. I still sometimes update my Twitter, but I'm not on Twitter anymore. Mastodon DMs are pretty terrible. I have my Discord—ChaelCodes. I have the YouTube—ChaelCodes. I have the ignored TikTok—ChaelCodes, and the Twitch. I'm very consistent in my branding, and it turns out nobody else wants my name.
[laughter]

JARED: That's convenient. There are a couple of other Jared Normans out in the world, so I have to get a little more creative.

CHAEL: [laughs]

JARED: All right, well, thanks so much for coming on the podcast.

CHAEL: Thank you so much. It's been great chatting with you.

JARED: I really enjoyed CHAEL's perspective on everything to do with working in the open, the benefits of it, how to get into it. I think there's a lot to chew on there. And even if you're not necessarily going to go out and start streaming yourself, consider how you can change the ways in which you work or in the ways in which your organization works so that you produce more evidence to build your career on that you are, you know, the software developer that you are because you're going to lose a lot of that when you change jobs.

This episode has been produced and edited by Mandy Moore.

Now go delete some…