# Google Prodcast Season Three Episode Ten

[JAVI BELTRAN, "TELEBOT"]

STEVE MCGHEE: Welcome to season three of The Prodcast, Google's podcast about site reliability engineering and production software. I'm your host, Steve McGhee. This season, we're going to focus on designing and building software in SRE. Our guests come from a variety of roles, both inside and outside of Google. Happy listening, and remember, hope is not a strategy.

—

Hey, everyone, welcome back to another episode of The Prodcast, Google's podcast about SRE and production software. I'm your host, Steve McGhee.

JORDAN GREENBERG: And I'm Jordan Greenberg.

STEVE MCGHEE: This season, we're focusing on software engineering in SRE. And today, we have two guests to tell us a little bit about the development of a piece of software inside of Google. It's a system called Maglev. But we'll get to that.

Our guests today are Cody Smith and Trisha Weir. Can you both introduce yourselves?

CODY SMITH: Yeah, I am currently the CTO and co-founder of a startup called Camus Energy. But before that, I was at Google for more than 14 years from 2004.

STEVE MCGHEE: Cool.

TRISHA WEIR: And my name is Trisha Weir. I am currently the manager of the Traffic Services Dataplane infrastructure, SRE department. What that boils down to is ProdGFE, DDoS, and a number of other services all around the front end.

I've been at Google for 21 years. I started as an infant. No, straight out of undergrad, much like Cody, I started in hardware ops and worked my way into SRE and really found my calling there. And I've been in various forms of production-adjacent since then.

CODY SMITH: Fun fact, by the way. Trisha referred me to Google. I would not have worked at Google if it weren't for her.

JORDAN GREENBERG: Wow, this is so fun.

TRISHA WEIR: Cody and I have known each other since we were teenagers because we actually worked together at the student-run ISP at UC Berkeley. It turns out that if your idea of a fun student job is running an internet service provider, you might be an SRE.

JORDAN GREENBERG: Oh, wow.

STEVE MCGHEE: That's awesome.

JORDAN GREENBERG: That is amazing.

CODY SMITH: Yeah, that was a great gig.

STEVE MCGHEE: When I was in high school, I had a friend who ran a teenager ISP as well. And I thought it was the coolest thing ever. And I did not get a job, but I totally wanted a job there. I thought it was the coolest. And, yeah, so began my descent into SRE as well.

JORDAN GREENBERG: There was a lot of physical work, too.

STEVE MCGHEE: Yeah. So today, we're going to talk about rebuilding core systems with reliability in mind, namely this thing called Maglev, which is a piece of front-end infrastructure. And we'll talk about it more in a bit.

But first, let's hear a little bit more about our guests. And we talked about it for a second just then, but what prepared you to become an SRE for either or both of you? You mentioned the ISP. Do you think that really did it? Was that the only thing that prepared you, or was there more?

TRISHA WEIR: I think the first part of the SRE, the ISP work, at least for me when I was really getting into it, was physical work. Cody and I, we showed up for the job and they would hand us these bales of network cable, and we would crawl under the buildings wiring them for Cat 5.

Some of it, I think, is this idea of the importance of getting the internet to the users, really did sink in. Working at that ISP, you cared about the users. You cared when their internet connections were down. They had a really bad day. And I think that prepped me for the focus of reliability as an importance. What about you, Cody?

CODY SMITH: Yeah, I mean, I think I was a computer kid. I really did not go outside much. I was just

always on the computer playing video games growing up, and was very entranced by the web when it first came out, and was building websites in '95.

STEVE MCGHEE: Nice.

JORDAN GREENBERG: That's awesome.

CODY SMITH: Working at ResComp was a really nice complement to my education in Berkeley. It was, in theory, a part-time job, but often, well over part time, especially in the summer. We were working 60-, 80-hour weeks, and just a whole huge variety of different things.

And that's kind of the life of an SRE. You don't necessarily work on the same problem for a long time in SRE. If you're making progress on the problem, if you eventually solve it, you move on to the next biggest problem.

JORDAN GREENBERG: I think they call that automating yourself out of a job. And it's every SRE's wish to be able to do that for themselves.

TRISHA WEIR: But it's also being a generalist. And my favorite way to talk about generalists is to describe them as stem cells, because they're the type of person you can drop in, and they become whatever is needed, but they have the potential to be any number of different roles.

I think, for me, really, what got me interested in SRE particularly was working at Google and being in hardware ops, because when you're out in a data center and something goes wrong, who's fixing it? Who are you talking to on the phone? It's the SREs. They were calm. They were competent.

Even around campus, I kinda describe them: when they walked, they walked in formation. The wind blew in their hair and it bounced. Their mailing list was literally called, The Cool Kid's Table.

And you saw this group of people, and you're like, I want to be one of them someday. They know. They can just figure anything out. They drop into a situation they've never seen, they figure out what's going on. They are the cavalry. That's an appealing mission.

JORDAN GREENBERG: What a lovely image to have of the SRE with the flowing hair. I mean, at Google, our SREs are like unicorns. So that's what I'm imagining now.

TRISHA WEIR: Walking in formation. Coming in.

JORDAN GREENBERG: Yes.

TRISHA WEIR: Explosions behind them, just like a movie.

JORDAN GREENBERG: Yes, exactly. So since it took a while, you got here, you saw these inspirational people who were at the cool kids table. They were SREs. Alias that now to what it was like when you were in search and traffic. How did these teams interact with each other? Who did what? Did they have as nice hair? All these good questions.

CODY SMITH: Yeah, I started when the teams were just starting to differentiate. So it was like production team back in the day, which then underwent a cell division and turned into search SRE and traffic SRE.

I had been working in a group, called cluster ops, and worked together in the Mustang War Room with a bunch of who soon be called SREs, who, after the experience working together in the war room, just picked up my desk and relocated it to their part of the office.

STEVE MCGHEE: You've been selected.

CODY SMITH: And I had a hard talk with my old manager.

So in the early stages, there was a lot of work that was moving back and forth and a lot of folks that were straddling the two teams. Some people were specializing pretty quickly and others were going back and forth, like me. And I had a lot of interest in the traffic stack personally, and so I took on projects like Maglev over on that side just, probably, because of my short attention span.

STEVE MCGHEE: And to clarify for our listeners, Cody, you were on the web search SRE team, and then, Trisha, you were on traffic team, which was about front-end infrastructure, load-balancing, and things like that. Is that correct?

TRISHA WEIR: Right. And we were the first, really, infrastructure SRE team and search was the first product SRE team. And I think GWS  followed pretty shortly thereafter.

I think the teams were really close. We sat together, we ate lunch together, we hung out after work together. And a lot of that was really helpful because we saw each other as one team. It wasn't like, oh, that's that team's problem. If it was their problem, it was our problem. It made for a really high-trust situation, which I think we'll get into later. But it was really helpful.

And there was a real cohesion. We all took our annual ski trip off-site together. We called it "the single point of failure day," because if there was a big food poisoning incident, it would really hurt Google if everybody got sick at once. I think the interactions were really positive and really, really close ties.

STEVE MCGHEE: That's awesome. So getting into the technical meat, the subject at hand, can you tell us what do we mean by front-end infrastructure? Don't you just throw a bunch of nginxes out there and call it a day, or is it a little bit more than that?

And then what is it that happened? What was the story like? We didn't want the thing anymore, so we're going to do something else. What was the story of Maglev's beginning?

TRISHA WEIR: So when we say front-end infrastructure, the joke is it's load-balancers all the way down. But it really is. Everything that has to happen, if you type in news.google.com, everything that happens, from you hitting Enter on the keyboard to your query getting to a news backend, or an ads backend, or a Gmail backend, that's all really front-end infrastructure and front-end networking. It's massively complex.

All of the things that allow us to have high reliability come with high risk of outage as well. We need to be able to load balance traffic across the globe. We do DNS-based load balancing. We need to be able to do failover between regions. We need to be able to absorb DDoS attacks. There's a whole lot of things that are all happening behind the scenes.

STEVE MCGHEE: And one thing about front-end infrastructure is that it's shared. It's not like everyone has their own stack. It's not like Maps has their own front-end infrastructure and then Gmail has their own.

TRISHA WEIR: Absolutely.

STEVE MCGHEE: Cool. So then, yeah, what was the problem? What did you guys face?

CODY SMITH: Yeah, right at the front of the stack, there's a thing called a network load balancer that's doing IP- TCP-level balancing of packets as they come in. And so it's keeping track of TCP connections and making sure that packets on that same connection always go to the same backend.

From the network load balancer standpoint, the backend is the Google front-end GFE. And we were using a device provided by a vendor in the realm of network hardware. The vendor's focus is on network hardware.

And they were pretty expensive. I think they were something like $20,000 a box.

JORDAN GREENBERG: Oh, wow.

TRISHA WEIR: And then the support contracts went into the millions for all of the devices, yeah.

STEVE MCGHEE: That's where the real money is.

CODY SMITH: Yeah, and that 20k would get you a gigabit of throughput. When we started the project, it was bi-directional, so it had to carry packets both inbound and outbound.

TRISHA WEIR: Getting on the DSR was my first project.

CODY SMITH: Yeah.

TRISHA WEIR: I remember going to the team picnic. There was a henna artist doing henna art. And we wrote "DSR or die" on my knuckles, like a knuckle tattoo.

JORDAN GREENBERG: Wow, I love that.

CODY SMITH: Yeah, "DSR" means Direct Server Return. And so you can imagine a packet coming in through the network load balancer and going through a GFE. And then when it goes back out to the user, it just goes straight to them and not back through the load balancer. And so this saves you on egress bandwidth and removes a big constraint in your load balancer capacity.

JORDAN GREENBERG: So there must be a reason why there are so many comments about these load balancers. Did something happen?

TRISHA WEIR: A lot of things happened, and they kept happening. There were some basic things, like we needed features that the vendor would just say, that's not a priority for us. We wanted to do GRE encapsulation to be able to allow us to do more advanced routing of the traffic. And the vendor would just say, we can't implement that on our devices any time soon.

The bigger issues were that-- well, actually, another growth issue is that they couldn't push very large configurations. And as we grew, our configurations got larger, and larger, and larger.

But the biggest issue is they were deployed in an active/passive pairing. And this was our redundancy. If one of them had a problem, you failed over to the other one.

But it wasn't a clean failure. The failover process caused these massive ARP storms. So you already have a failover happening, and then you have an ARP storm on the network because of that. And we brought all of these concerns to the vendor, and they really said, it's just not a priority.

CODY SMITH: Yeah, they were very focused on features. Their other customers just really wanted lots and lots of Layer 7 features that we didn't really care about. We were just bulk trying to move traffic in and out. And so we were weird compared to the rest of their users.

STEVE MCGHEE: And so the next thing was you just tried it.

CODY SMITH: Yeah, Eisenbud and I. I should explain, actually, a little bit of context.

We had this mission-control program where engineers would come over from pure dev teams to join SRE for a rotation of six months. And then if that had worked out well, they might stay.

So Eisenbud was one of the few folks in the team that had that product-development experience from before coming over. And she had reasonable confidence that we could do this, and the intuition about how to do things that I did not have because I had come straight from undergrad into SRE, basically.

So with air cover from our manager, Jinnah, who was also consternated about challenges with this vendor, we started breaking down the problem into components and splitting them up. We made a list of-- the load balancer is going to have to have these seven subcomponents or something. And then she and I just took turns grabbing the next-highest priority component from the list.

STEVE MCGHEE: Yeah.

CODY SMITH: And that went, actually, much more quickly than we would have expected. I think of a network load balancer as a scarily complex thing. But I think it took three to six months between starting coding and getting the first live traffic onto the software stack.

JORDAN GREENBERG: That seems pretty fast.

STEVE MCGHEE: That seems fast.

TRISHA WEIR: It was ludicrously fast. There was, maybe, two months between-- I think you had a prototype within two or three months, if I remember right.

JORDAN GREENBERG: Wow.

TRISHA WEIR: We were running a little bit skunkworks. So that was important because you needed to-- I think important context for this is, we had just had another project on the team that had gone for a very long time and never successfully launched. It was an internal tool. And due to a number of scope-creep issues, it never launched and was scrapped after a year.

And so that was a challenge to credibility of software development and SRE. And so rather than starting this one with bold announcements, that we were going to do another big project, that we were going to try this again, it was underground skunkworks.

And I think that really owes a lot to two things-- one, to our manager, Jinnah Hussein. He saw these people and said, I believe that these people can get this thing done. And if they don't, it's on me.

And then, to the engineers, trusting the manager. Because if you're an engineer and you work on a skunkworks project or you work on something that's pretty ambitious, and it goes wrong, that's not great for your perf. It's not good for your promo route. It's a risk in your career. And so they needed to be able to trust that the manager would have their back if something did happen, and would make it OK for them. So I think that was a big key.

STEVE MCGHEE: I work with a lot of customers and advise them on what SRE even is, and how to, and stuff like this. The idea of just building a thing in skunkworks, I think, just doesn't come up in many enterprise environments. And I think the high trust you're talking about, and just the skunkworks, and the ability to come up with something reasonably quickly is pretty awesome.

So I have a question here, which maybe is not relevant, but it was basically like, how was SRE involved in the design and implementation of it? But I think what you're saying is, I mean, you and Eisenbud just did it. You just did the design and implementation straight up. There wasn't a round of reviews, and an architect sign off, and a cost-analysis spreadsheet or anything like that. Is that accurate?

CODY SMITH: No, we did write a design doc. And it had diagrams of how we were going to handle these esoteric things like packet fragmentation. And we had some novel stuff in the design, like consistent hashing function for backend selection so that even under SYN flood attack, we could do reasonably accurate matching of incoming connections to backends.

And I remember Urs actually commented on the design doc. He replied to the email thread and said, I see you are not planning to respond to ICMP requests over a certain rate limit. But people ping Google a lot, and they really want to get those ICMP responses back. So please just answer as many of them

as you can. It's important. Yeah.

STEVE MCGHEE: Yeah, everyone is always pinging Google just to make sure the thing works.

CODY SMITH: Yep. I do it too, I mean.

STEVE MCGHEE: That's awesome.

TRISHA WEIR: There were some big shifts as it was going. You were talking about SRE involvement. Some of it was, I mean, it was input from the broader SRE team on how this was going to go.

At the time, we had a really strong distinction between control racks and production racks. Control racks in a data center, they're fixed in place. They usually have rack-mounted machines. Things don't change in them very frequently. They're often specialized hardware devices and--

STEVE MCGHEE: They're very special.

TRISHA WEIR: Yeah, our previous third-party hardware network load balancers were all racked into control racks.

And when we first started looking into this, we knew we wanted it to run on commodity hardware, but we were thinking of having that regular commodity hardware machine in the control rack. And there was a long tangent into, how would this work?

But we had a long tangent into how this is going to work. And I will say, one of my own concern is coming from hardware ops. I'd been in hardware ops. I'd repaired machines in the data center.

Anything in the control rack, that didn't come in through the system where you just got a queue of machines to fix, that was a special case. They didn't always have the parts. It was a one-off.

We started exploring what it would take to get these to just be production machines in prod, in the cluster. So if you've got a Maglev running on a production machine and it fails, turn it up on the one next door, turn it up on the next free production machine. It's not as much of an issue of, file a ticket, get someone to repair the hard drive on that machine, and reinstall it.

And so I would say that was wisdom of broader SRE. And I do think that was broader SRE input into Maglev in particular.

STEVE MCGHEE: In terms of timeline, this is also post-Borg, I believe. So it wasn't like, find a space in a spreadsheet for a machine. It was like type button, hit enter, get machine. It was totally automated in that.

CODY SMITH: Yeah, I remember stealing a fair number of machines from Borg for Maglev provisioning when we were doing the cut over.

So if you don't mind me switching gears just a little bit, can you tell me about turnup-- not the root vegetable. What that is and how it rolled out?

CODY SMITH: Yeah, the thing we were the most nervous about in the early days, we tried doing this thing. We took a piece of open-source software, I think called Quagga which was a BGP client, and wrapped it in our own control layer so it would interface with protocol buffers, and Stubby, and so forth.

So it would connect to the cluster router and establish a BGP session and say, for this VIP, I am the next hop. And the routers were then configured to allow that. And so we could start migrating VIP by VIP traffic over from the existing vendor device onto these Maglev servers.

And we knew from doing load testing of them that the throughput was actually fairly limited. The Linux kernel was not great at that time at distributing heavy network workload over multiple cores. And so these machines were pretty beefy, but they were trying to do all the work on one core. And so we had a limit in the hundreds of thousands of packets per second per machine, like 200,000 or 300,000, whereas line rate for 64-byte packets would be 1.5 million.

STEVE MCGHEE: Yeah.

JORDAN GREENBERG: Oh, wow.

STEVE MCGHEE: So just to clarify, when you're saying you went from VIP to VIP, so this is just a way to partition traffic. We don't have to get into all the details of what it is. So instead of taking all of google.com all at once, we take a logical subset of it that represents map tiles, or something like that, or SSL traffic for email or something like that, and just say like, just this slice, we're going to mess with today, and see if it works. Is that about right?

TRISHA WEIR: Google Toolbar, if anybody remembers that, toolbar queries were often one of our first things because it was an older product. It didn't get used as much. It was lower priority. And so we did

have some VIPs that were designed for traffic that was more tolerant to risk, I guess is how I would put it.

STEVE MCGHEE: That's important. Yeah, being able to have something like that established up front helps a lot when you want to do an experiment like this. Because if your plan is like, we're just going to turn it off and turn it back on again and hope, not a great plan.

OK, so you rolled it VIP by VIP. What next?

CODY SMITH: We started with  risky VIPs and got good results from that. Everything seemed like it was working fine. And we were doing some amount of redistribution because these machines weren't as fast as the vendor device. And so we couldn't just do a one for one cutover.

But we just gradually worked our way up to the more crown jewels VIPs that web search and other super important products used in different clusters on different types of hardware, because each hardware had, potentially, different weird constraints, and special kernel configurations, and things like this. And just have all the monitoring active. During this cutover, the monitoring just hits the VIP. It doesn't actually know what's implementing the VIP. And so those probes to the front-end infrastructure would tell us if something had really gone off the rails.

STEVE MCGHEE: That's awesome.

TRISHA WEIR: We had a lot of air miles to get. We would have schedules like, OK, we'll do this much and then we'll get air miles on it. We'll bake it for this many weeks. Then we'll do this much more.

One of the things we always try to do is to get every type of diversity of traffic, of type of machine. It's less true now, but back then, certain regions of the world were much heavier on video traffic versus ads or search traffic. And so the traffic dynamics were different in different regions. You would have places with high packets per second, places with high queries per second. And we really tried to get a broad distribution of having air-miles clusters or test clusters all across the world.

STEVE MCGHEE: Yeah, awesome. Heterogeneity or diversity in your workloads is really important when you're trying to satisfy something like this.

JORDAN GREENBERG: Yeah. And so I'm imagining that, as you created these environments to be able to try these things in, did you find any good bugs while you were testing this stuff out?

CODY SMITH: Yeah, there was one bug that I still think about, probably, far too often. It was in my code.

JORDAN GREENBERG: It haunts you?

CODY SMITH: Yeah. I wrote the kernel interface. So we used the same interface that tcpdump uses to get packets from the kernel, and the packet parser code that goes together with it. And I was paranoid that there would be a bug in here somewhere.

And so I wrote a fuzz tester that clamped on to the packet parser and literally ran a giant MapReduce with this code to process a trillion packets. Huge job. And it said no bugs. And I was ecstatic.

But then the first time we got a packet that was larger than the snap length on the buffer that we had given to the kernel, the snap length and the packet length were different and we were reading the wrong one in our code.

And that's OK. For most of the ring buffer, you read into the subsequent packet. But if you're right at the end of the ring buffer, you read out into unallocated memory and then use segmentation fault.

And so we had done a turn up where some local loopback traffic was 64-kilobyte MTU, and it was getting truncated in the buffer and crashed. And we figured out the problem right away. It was a very easy fix, but very embarrassing because it was like five lines of code before the fuzz tester plugged in.

JORDAN GREENBERG: Oh, wow.

STEVE MCGHEE: So I get the sense that this wasn't Rust, or Java, or some fancy memory-managing code language of modern times.

CODY SMITH: Yeah, it was C++.

STEVE MCGHEE: Nice. Respect.

CODY SMITH: Yeah.

TRISHA WEIR: I think the other issue we started to run into as we were doing rollout was, we started saturating the rack-top switches. We'd considered the availability of bandwidth there as a concern. But it wasn't something that was built in as a concern to any of the systems that did auto replacement.

And when we began being placed on racks that had other neighbors that were also high-bandwidth

users, we started having saturation problems. And so I remember having to work with-- I think, Isidore was the tool at the time. Cody might remember.

STEVE MCGHEE: Yeah.

TRISHA WEIR: Isidore-team to write and implement a new constraint so that when it was allocating machines, it would check for that kind of diversity as well.

STEVE MCGHEE: So would you say that maybe it wasn't perfectly plannable, and instead, your teams had to adapt to emergent behavior that was resulting in this change that you made on a very complex system?

TRISHA WEIR: I think that that's fair. I do think that it's the sort of thing we think about now. It was a brave new world. And we had, given the sort of ambient knowledge of the industry at the time, we planned it decently well, but it always could have gone better.

STEVE MCGHEE: I mean, we're here today. I'm pretty sure Maglev is handling several hundreds of packets per nanosecond. I don't know. Some sort of number. OK, so--

JORDAN GREENBERG: There's math involved.

STEVE MCGHEE: --with regards to SREs who aren't at Google, that's what most of our listeners are on this podcast, is this story too ambitious? Do you need to be the size of Google to do something like this? Not necessarily this exact task, but this process of replacing a thing with software, and iterating, and building it out, and seeing if it works. How would you advise folks out in the world to take this story? What should they learn from it?

TRISHA WEIR: I think there's always a balance to be struck with knowing, is this something I should build myself or can I make my use case fit existing tooling? Because you can move faster if you stand on the shoulders of giants. And so I think one thing to develop would be that sense of intuition about, is this something that no existing tooling is going to do, and do I need all these features?

One thing Cody and I were talking about before we were on air today was the idea that there was a list of features that our previous hardware device could do. And Cody and Eisenbud, they just dropped a lot of them. They're like, we don't actually need that special feature. We don't need that special feature. What we do need is these particular set of features that nothing else on the market is giving.

And so before starting a project like this, to decide if it's a good use of your time, I would say build that intuition and that awareness. Everybody wants to build something new, and no one wants to do maintenance is a Kurt Vonnegut quote that I really like. People often want to build new things. Building new isn't always the answer, but having the judgment about when it is the right answer versus not is something that will really help you.

STEVE MCGHEE: Nice.

CODY SMITH: Yeah, I would say, I think, the time pressure was really useful in the context of the thing we needed to build. It didn't actually prove to be that complex, and a big part of that is the brutality with which we kept things simple. The whole thing, not counting the BGP client, the code base was, I think, 2,000 lines when we launched.

STEVE MCGHEE: Wow.

CODY SMITH: So it's just not that much code. And the time pressure of like, this is a skunkworks thing, we have to demonstrate a result before we get canceled, was really useful because it made a lot of the decisions about, do we need to support this feature on launch day or not very simple. If the answer can, by any means, be no, it should be no.

JORDAN GREENBERG: So what advice would you give today's kids who are working at the ISPs, figuring out who they are, figuring out who they're going to be working with for a long time for the next 20 years at some really huge company that may or may not even exist yet? What advice would you say to them about how they can improve the world's internet and make it bigger, better, faster, or stronger?

CODY SMITH: I think early in your career, trying to find roles that have a big container around them-- there's a lot of different places you can go inside the role, and you're not going to be doing just one thing over and over again, but lots of different things that all have to work together-- is really great for making yourself into a stem cell that could go out and get a tech job in a bunch of different ladders at a big company or become an entrepreneur and go do something you've never done before at a startup. It sets you up. It gives you that breadth of skills so that you have more choices later in your career.

STEVE MCGHEE: Nice.

TRISHA WEIR: So I am a big believer in the idea that SRE has an above-average need for psychological

safety. We are in the middle of an outage. We need to feel safe, suggesting a very wild solution, and knowing that your coworkers will not say, that's ridiculous. It would never work that way. I can't believe you didn't know that. Because sometimes, the wild idea is the thing that fixes it.

That's true of SRE, but it's also true of anybody who's innovating. You need to be able to innovate, to have a wild idea, to take a chance and know that if you fall, your boss will catch you, to know and like the people around you.

So much of, at least, my education was focused on, you need to write code, and you're going to write it all by yourself, and you need to be heads down and get really good at this. But everything I have seen happen huge at Google has resulted from massive collaboration. And a lot of that comes down to being able to work with other people, being able to integrate their ideas, to respect them even when you disagree.

So I would advise, find people you enjoy working with, because if you're not enjoying your time, if you're not feeling safe, if you're not able to be your whole self around them, you're not going to bring your best ideas, you're not going to be able to get other people to work on your ideas, and you're not going to go as far. And it's just not as much fun.

You don't have to be best buddies with everybody. You don't have to hang out on the weekend. There are plenty of people on that team who I adore, and we see each other once a year now. We're not besties, but there is someone I think of as someone I trust, and like, and respect. And I think that that aspect is really important.

JORDAN GREENBERG: That's awesome.

STEVE MCGHEE: I have to say, it wasn't until I left SRE and Google that I learned the phrase, "stay in your lane." And I thought it was pretty prescient that that wasn't a thing inside of Google, or inside of SRE, at least. No one was told to stay in your lane. Because it's like, these lanes, there's so much stuff. Go ahead. And it just works better that way. Totally.

JORDAN GREENBERG: Yeah, I would agree.

All right, well, thank you to both of our guests, Cody and Trisha. Cody, where can we find you on the internet?

CODY SMITH: Yeah, the company I co-founded and work at now is called Camus Energy. We are

camus.energy. And if you want to email me, I'm just Cody@camus.energy.

JORDAN GREENBERG: Perfect. How about you, Trisha?

TRISHA WEIR: I am at Google, so you can email me at Trisha@google.com. T-R-I-S-H-A.

JORDAN GREENBERG: Perfect. Well, thank you, everybody, for making this episode of The Podcast possible, and have an awesome day.

STEVE MCGHEE: Thank you.

CODY SMITH: Cool. Thank you so much.

TRISHA WEIR: Bye.

STEVE MCGHEE: Bye.

—

[JAVI BELTRAN, "TELEBOT"]

JORDAN GREENBERG: You've been listening to Podcast, Google's podcast on site reliability engineering.

Visit us on the web at sre.google, where you can find papers, workshops, videos, and more about SRE. This season's host is Steve McGhee with contributions from Jordan Greenberg and Florian Rathgeber. The podcast is produced by Paul Guglielmino, Sunny Hsiao, and Salim Virji.

The podcast theme is “Telebot” by Javi Beltran.

Special thanks to MP English and Jenn Petoff.