Google Prodcast Season Three Episode Seven

[JAVI BELTRAN, "TELEBOT"]

INTRO

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.

STEVE MCGHEE: Hi, everyone, and welcome to another episode of the Prodcast, Google's podcast about SRE and production software.

I'm your host, Steve McGhee.

And as always, we have our co-host, Jordan Greenberg.

JORDAN GREENBERG: Hey.

STEVE: Hey, Jordan. How's it going?

JORDAN GREENBERG: Good. I'm excited.

STEVE: All right.

So this season, we're focusing on software engineering in SRE.

And today, we have two guests to talk about how that's done in their industry.

Specifically, these two come from retail and from gaming.

These two both share two traits that we're interested in.

One is what I'm calling real-timeyness, and maybe less obvious, the need to move or migrate between platforms, like onto cloud or between clouds or even between internal software delivery systems.

So let's have our guests, Jordan Chernev and Scott Bowers, introduce themselves.

JORDAN CHERNEV: Hey, everyone.

My name is Jordan Chernev.

Thank you for having me on the podcast.

A little bit about myself-- about 20 years of industry experience across both enterprise and startup cultures.

I'm a former practitioner.

Most of the organizations that I've been a member of have been either anywhere from 50 to 200 in size, all the way up to 20,000 multinationals.

I think of myself these days as a business executive with a focus on technology.

Primarily, I'm more of a senior engineering and product leader with a specialization in data, SRE developer experience, tech transformation, hypergrowth.

JORDAN GREENBERG: Thank you very much, Jordan.

How about you, Scott?

SCOTT: Hi.

Yeah, I'm Scott Bowers.

I work at Gearbox Software, who are most well-known for the Borderlands game franchise.

And I started there just out of college, so I've been there for about 10 years now and worked my way up through classic IT into our SHiFT platform, which is what we call our online game engine.

STEVE: Awesome.

JORDAN GREENBERG: Hmm.

And that's where I get my loot codes, right?

SCOTT: Yeah.

Yeah, I think for us, they're golden keys for the most part.

JORDAN GREENBERG: OK, OK.

All right.

Thank you very much, guests, for introducing yourselves.

Really appreciate it.

What is it like to be an SRE on stuff that people are using to unwind, whether it's in shopping or games?

SCOTT: [LAUGHS] For me, in games, it has been the right mix of responsibility and excitement.

We get to have a lot of people using our platform each day, but that also means that I stand to ruin their leisure time if I do things wrong.

STEVE: No pressure. JORDAN GREENBERG: True.

SCOTT: Yeah, yeah.

It's exciting when we get things right, and a different kind of exciting when we get them wrong.

JORDAN CHERNEV: Yeah, I'll plus one that sentiment and comment.

Obviously, shopping, most people want to have a seamless shopping experience.

The way that we did this at Wayfair was supporting thousands of services, especially for big gameday-like events like Way Day, Cyber 5.

Those were across hundreds of services, hundreds of teams.

They all had the one ultimate goal-- consistent and delightful experience for everyone who is trying to shop online.

Excitement is an understatement.

There is a lot of preparation.

There is a lot of planning, testing, making sure that everything is going to scale.

And there is always a little bit of luck in all of this.

I don't think we can understate that.

You can prepare as much as you want.

But I don't know, maybe some small service could potentially introduce too much excitement that you didn't account for.

STEVE: (SARCASTICALLY) And I don't suppose that either of you, in retail and gaming, ever deal with spikes of traffic during certain times of the year, or around events or anything like that?

I'm guessing that never really comes up?

SCOTT: (SARCASTICALLY) No.

We haven't ever had any trouble with a game launch.

STEVE: Oh, no? OK, that's good.

JORDAN GREENBERG: Certainly not.

STEVE: It seems easy.

SCOTT: We definitely have spikes, and we have definitely struggled at times with designing for those, and then also running an efficient infrastructure outside of those.

JORDAN CHERNEV: Yeah.

From our perspective, you do get the spikes, sometimes are anticipated or unanticipated.

One of the more interesting examples that comes to mind is sometimes you may introduce an innocent-looking header to a web page, which increases the load on a somewhat critical path that you didn't notice it was critical at that point in time.

By 100x it goes up, somewhat triggered internally.

But yes, a lot of experience with marketing, email campaigns, blasts that are somewhat scheduled.

Those are ones that you can prepare for, but it's a combination of the two.

STEVE: Yeah, marketing is real, man.

Turns out it has real effects.

Not a joke.

JORDAN GREENBERG: You mean it works?

STEVE: I'm afraid it does, yeah.

JORDAN GREENBERG: Oh, no.

STEVE: Turns out.

JORDAN GREENBERG: OK.

STEVE: So in my experience as an SRE, one thing that we talk about a lot is SLOs, and SLOs tend to be a thing that we focus on on is the thing working or not?

I'm curious if you guys-- do you use them at all?

And how does it relate to a service that you can see it's exact "is it working or not," in terms of money and customer satisfaction?

Do these SLOs align with how the business is doing?

And if not, can that be fixed, or is it a fool's errand?

Is this a good idea, or what do you think?

SCOTT: Man, that question kind of hits me right in the gut, because our first broad pass at trying to apply SLOs to our most important services, we were kind of engineers about it.

And we were thinking more about, what's the service doing, and not as much about, what's the user experience?

And that's always the actual important thing.

Like Jordan said, the goal of all of that stuff at the end is the consistent experience for the customer.

And yeah, having our SLOs currently a little bit misaligned from that, we can feel it.

So we're in the middle of an effort to redefine and realign some of those, so that what the player is seeing and experiencing in-game is the thing that our objectives are designed around.

JORDAN CHERNEV: Yeah, we've thought about those a little bit differently.

It's perfectly OK for teams to have their engineering or private SLOs that basically tell them the health of the components of the services.

But the ones that are considered more public, or basically end-user-oriented or business-outcomes-specific, you do want to have the healthy combination of the two.

Because when you go to an operational readiness overview for your systems, yes, let's take a look at the engineering-specific ones to see if we can drive consistency from the technology standpoint.

But obviously, for the ones that the business cares about, you want to make sure that you have nice

mental connections between this is what exists for the service; this is how it ties to a specific metric or a KPI that we have defined.

This sounds easier set out as opposed to done in practice.

Not all services will be able to have that direct link to a specific business-level KPI.

Maybe your top or most critical services would.

But the deeper you go into the back end, the harder that gets to be defined, especially if you have shared services that you're providing all for the entire organization.

How would you be able to quantify that or tie it to a specific business metric?

So there are some interesting twists and turns in terms of defining those.

But yes, it's a very useful method for you to try to communicate and portray the value add or the potential impact to a non-technical audience within your organization.

STEVE: OK.

So given all that, specifically in both your industries, like we said, you're kind of real-timey, as well as you're direct-facing to customers.

So millions of people can see when stuff goes down.

So I'm curious if you deal with SLOs on a per-minute basis, or more so per-second?

I don't know.

How fine-grained can you get with this stuff?

Do you know when your system is down for 30 seconds or is it more like-- does it matter, I guess is what I'm getting at.

Is it worth the investment of going to that level of granularity?

SCOTT: I think in our case, especially for our biggest couple of titles, we have designed ourselves into a spot where it does matter.

Second-level impact is noticed by every player online for a given title or, sometimes, depending on the nature of the impact, all titles using our platform.

So yeah, whether we know proactively or after the fact, we can absolutely see a second-level impact to our most real-timey service being our multiplayer lobby and messaging system.

People are in a lobby together.

They feel if the thing keeping them together breaks.

So that is definitely a thing that--

JORDAN GREENBERG: That's true.

STEVE: Lag is real.

SCOTT: --we pay a lot of attention to.

And we actually redesigned most of that multiplayer lobby system in the last year, just for better observability, with the goal of hopefully a truly defined objective as opposed to just a fuzzy keep-the-players-online.

JORDAN GREENBERG: The gamers are thanking you.

I'm "gamers," but the players are thanking you because a lot of games have evolved to be very social.

And for services where you have to be available when multiple people are available, people don't want to think, oh, the thing that I want to do with my friend isn't working right now.

They don't want to see that.

It makes them very sad.

SCOTT: Yeah.

JORDAN GREENBERG: So I'm sure everybody's excited about that.

SCOTT: Yeah.

I've joked with my wife that video games, for me, at my age now, are kind of just an excuse to hang out in a phone call with my buds.

So if we don't have that excuse, we don't catch up.

JORDAN GREENBERG: Mm.

STEVE: That's right.

And how about in retail, Jordan?

Do you guys have a similar kind of situation?

JORDAN CHERNEV: I think it's a little bit different.

Maybe it's slightly more relaxed from the perspective of end user experiences.

For better or for worse, I think most shoppers online tend to be a little bit more sticky, especially if they experience a transient or a temporary hiccup in terms of availability or performance.

Many people are used to, hey, I have to go and buy tickets for a concert, and I have to keep pressing the button just so I can get it.

So the mental model there somewhat relaxes the expectations for the end user.

We were capable of detecting and seeing issues as they were happening.

I don't think it was as pragmatic or practical for us to try to detect them at the second level.

But maybe at a minute, we will definitely start seeing more real business impacts.

And again, it's not like, hey, I had a brief minute of an outage; you're going to see a massive shift.

But obviously, the more that the minutes start stacking up, that's when you start seeing a little bit more of a profound softness in experiences.

STEVE: Yeah, for sure.

JORDAN GREENBERG: Mm.

So if that's happening, how do you communicate with the users of the product?

When things are going sideways or pages won't load, how do you say that to them?

JORDAN CHERNEV: It depends on the outage, I would say.

I think there are many different methods.

Each business tends to maybe choose its personality or persona that it displays.

You basically create, I don't know, a maintenance or a page.

You obviously want to communicate back with your customers via email after the incident has wrapped up, and basically communicate to them what happened.

Here's the story.

Maybe that makes even for a good blog post after the fact, around providing specific visibility.

Maybe you send them a marketing coupon or a code, basically trying to attract them back, making sure that they have a pleasant, good experience, if you will.

I think those are the main approaches that we have considered over the years.

That's how we describe it.

There is, as I mentioned earlier, there is a certain level of halo or stickiness.

So even if someone is experiencing something that happens for a brief period of time, they usually come within the same hour, because they're still actively online and shopping.

And you could actually see, hey, the revenue is somewhat-- it was a little bit soft.

Now it's coming back and then some more, which basically reattracts the people who are actively looking to shop.

So the more sophisticated that you get with your business, the more visibility and methods you do.

But in terms of communication, it's primarily follow up around either the different channels around text-based notification, push-based notifications, emails.

Yeah.

SCOTT: We actually, at least for a planned outage, we have a handful of ways that we can try to get that information to our players.

But for the more interesting case of an unplanned outage, we don't have a great way, aside from our SHiFT status Twitter page.

And even then, getting from, hey, we have the technical signals that there's a problem, to there's an engineer who's validated, yes, players are impacted, to getting a post onto the Twitter, there's a lot of time there for a gamer to think, hey, this should work and it doesn't, and the status Twitter says everything's fine or hasn't had a post in a day.

So that's something where we want to get better.

And we're hoping to use something like status page, which is something I'm a fan of just from various sites or tools that I use have that, so it's in my brain as something to type in.

But yeah, that is a challenge for us, if we have an unplanned outage, to communicate that proactively and then also to send the all-clear.

We can use the Twitter for that, but we're hoping for a better and slightly more technically-involved solution.

STEVE: Cool.

So given all these things that you guys have built, and your teams have been able to develop in terms of communication and investigating things, I have a broad question which is, how did you get this all to be invested in instead of the next game or the next feature?

So sometimes we call this shifting left, shifting reliability left.

How do we get this type of engineering done in an organization?

Is it, the ops team files a bunch of bugs and complains a bunch?

Is it writing postmortems?

Is it just stepping in and rolling up your sleeves and writing code?

How has it worked in your experience, in terms of getting this stuff more suitable for this large-scale, real-time, customer-facing stuff?

JORDAN CHERNEV: I think reliability is actually a team culture component, first and foremost.

In order for it to work, everybody has to be invested in that to be successful-- what I mean by everyone, the target personas are not just the engineering teams.

There has to be also investment from the business, investment from the product teams.

Basically having those shared incentives around, this is what the end user experience looks like in aggregate to the people who are using our product.

So I think by starting to evolve and shift the culture to be a lot more focused, or at least mindful about reliability and how that may impact the actual product that the organization is delivering, I think that's probably the first and most important starting point.

STEVE: Cool.

Scott, what do you think?

SCOTT: I'm lucky that I joined the team at a time when they were redesigning the entire platform from what we have internally called platform V1.

They were designing platform V2 with reliability and observability as the goals.

And I think that is due to the technical leads on the team, before I joined them, seeing the scale that the business wanted to achieve, and seeing that what they had built at that time was not on the same track.

And I think that along with some of our team culture of make-on-call-not-suck, those two things came together so that the right people were invested enough in the idea of reliability that, man, multiyear effort to design and then migrate to our new platform was authorized.

STEVE: Nice.

Yeah, it always comes down to scale, I think.

It feels like that's the thing that pushes you over the edge.

It's like, would you like to make more money today?

OK, this is what we got to do, turns out.

OK, cool.

Let's shift gears into migrations.

So Jordan, take us away.

JORDAN GREENBERG: Yeah, what is a migration?

Why do you have to do it?

SCOTT: That is a fun semantic rabbit hole, and I think that my personal definition of migration, so that I can get my dang job done, is something where we're changing a foundational or fundamental piece of a feature that we serve.

And it's a one-way change, and there is a point of no return that we have decided on together as a team.

And then there are so many soupy details in between, and so much of that plan always ends up changing.

But yeah, for me, it's a foundational change that is ultimately one-way.

JORDAN GREENBERG: What do you think about migrations, Jordan?

Do you like them?

Do you hate them?

Are they necessary?

JORDAN CHERNEV: I mean, the running joke in the community is that once you're in the platform or the SRE or a dev ops space, the migrations don't ever stop.

They're just a fact of life.

They are a necessary way for you to either drive innovation, positive outcomes, or even pay down technical debt.

So they are good ways for you to provide a good healthy balance to where the organization wants to go next.

They are always running, never-ending.

Once you complete a migration, another one starts.

Or maybe there are multiple migrations happening at the same time, which sometimes can have potentially conflicting schedules and agendas.

So it's really about constructing the right sequence of timelines across different teams.

So it's a fun place to be, especially if you like to be somebody who is really organized.

But as mentioned earlier, it's generally a positive thing that just never ends. JORDAN GREENBERG: Oh, yeah.

OK. So this just in-- SREs are migratory birds.

No better.

STEVE: [LAUGHS] Nice.

Specifically, how does this affect you with this real-time stuff?

Because it seems like a migration can be a real bummer.

And if it goes the wrong way and people notice when things are down for a couple of seconds, that seems like a bad combination.

So any top tips on this?

How do we allow ourselves to get over this hump, and not completely have our users suffer along the

way?

SCOTT: Oh, man.

For my team, doing our platform V1 to platform V2, going from one big cloud account to a spread of them and trying to follow a bunch of best practices along the way, we had to accept that we were going to just cause impact.

Sometimes there were some services where engineering for zero downtime was not realistic or cost-effective.

And in those cases we got lucky, to some extent, that in the gaming industry, regular downtimes are accepted.

Gamers know from big companies like Blizzard and Steam for years and years that have had weekly maintenance windows that it's part of the gaming space.

So we don't have a regularly scheduled maintenance.

But we, when we know we're going to cause an impact, try to give a lot of notice ahead of time both in-game and on the social channels.

But for a lot of our services during that migration, we made use of a pretty neat and relatively simple trick to work around the fact that we don't have an A/B or a blue/green deploy infrastructure setup.

We just had old platform, new platform.

And so we used weighted DNS records with CNAMEs pointing either way so that we could have just a percentage weight there.

And it was zero pointing at the new stuff to start with.

And for things where we didn't think we needed downtime, we could set that to send 1% of our production traffic to the new platform and see what happened.

And we knew we were risking 1% of our players' experience, but it gave us the scale to feel confident in going up to 5% and beyond.

STEVE: Yeah, gradual change, right?

SCOTT: We tried and did not always succeed.

JORDAN CHERNEV: Yeah.

Yeah, I think Scott did a pretty good job of enumerating most of the strategies.

Obviously you have options like double writing or dual writing if you're trying to set up or stand up different data platforms.

This is where a message buses can become really your friend, especially if you're in a situation where you have to migrate and validate the data before you do the actual cutovers.

Those are the trickier and the longer-running ones, if you will.

But yeah, blue/green, draining, rolling strategies, hard cutovers.

For some services, maybe it's not even worth the time of coming up with a zero downtime strategy.

For instance, I have an internal tool that is being used by two teams on the business side.

They only use that service during the hours of 8:00 to 5:00 every day.

And maybe I can just do a clean or easy cutover, from a technology investment standpoint, during the hours of 6:00 to 8:00 PM when they're not around.

So it really depends on the actual service, the context, and how much impact, from business perspective, you're willing to tolerate.

It's basically a trade-off.

JORDAN GREENBERG: Nice.

So just thinking about this, when do you engage SREs to help with migration?

JORDAN CHERNEV: I think you have to start during the early socialization stages of any kind of product, because these activities are usually as a part of the actual product life cycle.

In practice, it is rather unfortunate.

Many people miss this, and usually SRE get invited towards the middle or the end phases of the migration.

People are somewhat used to that, which is not necessarily a healthy pattern.

But ideally, the sooner you can get involved as a group, the better.

This is where good relationships with folks on the business side or the product team could actually help you hear about some of these developments a little early as a group.

I don't think it's going to be like a-- I don't know.

I think the way that I've seen it, in my experience, it's about maybe 10% or less of our engagements usually have SRE as an active party from the get-go.

JORDAN GREENBERG: I think that they did that on purpose across all companies, because they're worried that they'll scare the engineers if you ask them questions.

I think that might be it.

Maybe turning on the lights scares them as well.

SCOTT: That matches my experience so far, that SREs have been a critical component of design later in the process than we would have preferred.

JORDAN GREENBERG: Right.

SCOTT: And I think my team of SREs is lucky that we've got a bunch of software engineers that want to implement observability and metrics and be cutting-edge in all of these ways.

But then, yeah, the tensions between moving fast and doing it really well have not always worked out in our favor.

STEVE: So, Scott, I think you mentioned the point of no return.

So in the migration, whether it's changing a platform, changing the implementation of an entire codebase, changing an infrastructure, what do you mean by a point of no return?

And is there always one?

Can't we just roll the thing back infinitely, or do we have to bite the bullet at some point?

How do you plan for this stuff?

SCOTT: I think my personal philosophy that I try to convince my colleagues of-- and so far, it's worked-- is that mainly for a cognitive burden and team sanity standpoint, it is just useful to have a point of no return where we can say, if we achieve feature parity up to this level in the new deployment of the service, then that's where we're done.

And I mean, I think technically, yes, we could always roll back or engineer our way out of the new thing, back into something very close to or exactly like the old thing.

So it's always going to come down to, what does the business demand?

But if there's room for engineering input, then yeah, my choice is always going to have a point of no return so that all of the people working on the thing can checkpoint and cache dump and move

forward from here, and not have to retain too much historical or tribal knowledge.

JORDAN CHERNEV: Yeah, that also matches my experience.

Whenever teams are creating game plans, we make it a point of specifying of-- and from here on, this particular step, this is the point of no return.

So once you cross the Rubicon there is no turning back.

You have to fight your way through the remaining issues that you might be observing in the environment.

And we feel pretty comfortable communicating that to the rest of the business, around the expectations that once we're kind of like, from this point, there is no turning back.

We just have to keep going.

So I think providing that level of transparency ahead of the actual migration, I think, is very helpful, because the business knows what to expect.

It's not perceived as a seamless or nonturbulent migration.

STEVE: Cool. OK.

We're running out of time here.

This has been going great so far.

I have one more question, and it kind of comes off of what you just said, Jordan.

Specifically, when people start using the cloud, often, one of the things that we talk about, and some of the things that the business has to think about, is that we're actually giving away a lot of control.

Or maybe we're giving away a lot of responsibility.

You can think of it either way.

How does this come up when it comes to talking to the business about this?

Is there a risk versus control versus time and cost and flexibility trade-off that you've seen be successful when it comes to, yo, we're going to use this crazy thing that lives in the sky now, not those machines that keep our buildings warm at night?

How does that go, in your experience?

JORDAN GREENBERG: The free heaters.

STEVE: They do heat well, that is true, for sure.

You can keep your sandwiches in there, and it's great.

JORDAN CHERNEV: It's a great question.

I think at this point in the game, the public cloud concept is more or less somewhat fairly well-understood from a business standpoint.

They know what to invest.

They know what kind of experiences that they're going to get.

There is a maybe correct or potentially correct perception that the public cloud is potentially a lot more secure than a private implementation.

There are some nuances and shades of gray to the statement, so please don't feel like it's as black and white as I just said it just now.

But a business sees that as a potential value add, because it helps them increase their overall security posture.

They see that there is a benefit towards mean time to market for new initiatives, or anything that is brand-new and born with cloud-native paradigms and shifts.

They see it as an opportunity for you to accelerate velocity for developer experiences, because most of the interactions that you have with the cloud are more or less API-based or API-driven.

However, the cloud comes with its own set of challenges and complexity around skill sets, so you obviously need to be able to navigate that.

Cost is a topic that the business is not necessarily fully aware of, the transition from CapEx to OpEx or how to actually attribute costs with the OpEx model to specific teams within your organization, as opposed to one big bucket who is responsible for it.

Those are topics around governance that usually businesses are not necessarily ready for.

Usually the business has a fairly decent understanding, given the level of maturity.

Right now, it's a little bit more mainstream for you to try to think about, hey, is the cloud too expensive in terms of margins?

Can I potentially save by going back to a colo, using a private cloud for specific workloads which are

not as bursty?

Obviously that's a specific business decision first, less so technology.

JORDAN GREENBERG: Well, thank you for talking about the cloud magic stuff.

I'm sure it's fine.

I'm sure whoever does cloud stuff is fine.

The last question we have for you today is, thinking about some of the people that you've worked with in the past who may have been on that IT team that you were on before, Scott, or other people that weren't SREs, could we train them to become SREs?

What are the challenges of that?

SCOTT: I think we definitely can.

That's actually been one of the more rewarding slices of my personal responsibilities lately.

Our classic IT team, whenever I left them to join the Spark online platform, I got to carry with me a lot of their roadmap and goals and then learn a whole bunch of new skills.

And the original idea was to back channel and teach them along the way, and I was too focused on how much awesome new stuff I was learning.

But now, both of our teams have realigned and a lot of their work is catching up to the style that we've landed at for the way we build our infrastructure, and then taken the reins from us in some ways.

So yeah, I definitely think that it's a skill set that can be taught to the classic ops folks and can be used to make their lives easier, I think has been the thing I've seen the most of.

JORDAN CHERNEV: Yes, it is certainly possible.

In my experience, it really comes down to the individual or the teams who are undergoing through that transformation.

Obviously, it's a bit of a personal journey of growth, reskilling, upskilling.

You should be trying to create a positive environment that encourages that type of transformation.

It doesn't happen overnight.

It's usually at least 6 to 12 months for somebody to go through that journey and be somewhat comfortable with the newer skill set and the responsibilities.

The good news is that some of the required fundamentals are still the same, meaning if you know how computers work at a fundamental level, at CPUs, memory, network, storage, those will translate very, very seamlessly.

I think, usually the biggest hurdle is how do we actually translate that to how public cloud providers have decided to put an API on top of those fundamentals, so we can actually learn how they actually work?

Knowing and understanding that, and having the actual skill set around specific frameworks, for instance, Terraform and maybe Golang, I think those are usually the areas that basically provide a step function in terms of ability or confidence level, in my experience.

STEVE: Cool.

Well, thanks to you both.

Before we finish off, do either of you have a final take, or maybe where people can learn more about you online, like if you have a blog?

Do people still blog? I think they do.

JORDAN GREENBERG: The socials.

STEVE: Or anything like that.

A social, as the kids say.

Anything like that you want to shout out?

SCOTT: Not for me.

I avoid all that.

STEVE: Good answer, good answer.

JORDAN GREENBERG: Probably for the best, honestly.

JORDAN CHERNEV: Yeah.

For me, you can find me fairly active in the platform engineering Slack community.

Obviously, feel free to hit me up on LinkedIn.

I try to do a weekly post on the topics of platform engineering every Friday.

So if you follow me on LinkedIn, you're likely going to see a subscription of my feed.

So happy to see you there.

STEVE: Thank you.

JORDAN GREENBERG: Thank you very much.

All right. Thank you, everybody.

Thank you to our guests, Scott and Jordan, for helping us shop and game.

We need to be able to have fun, especially if everybody is fighting fires during the day, keeping our other services alive.

Have a great day, everybody.

STEVE: Bye.

JORDAN GREENBERG: Bye.

[JAVI BELTRAN, "TELEBOT"]

JORDAN GREENBERG: You've been listening to Prodcast, 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 Prodcast is produced by Paul Guglielmino, Sunny Hsiao, and Salim Virji.

The Prodcast theme is "Telebot" by Javi Beltran.

Special thanks to MP English and Jenn Petoff.