Google Prodcast Season Three Episode Two

ROBOT: You missed a page from telebot.

STEVE: 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.

[ELECTRONIC MUSIC]

Hi, everyone, and welcome to season three of the Prodcast.

I'm Steve McGhee, and I'm your host this season.

Today we have a special kind of in-between episode.

Since last season was about the different stages of being an SRE in Google and outside of Google, we now have a guest today.

We have Healfdene Goguen and he is an SRE inside of Google.

And he actually started as not an SRE.

He started as a software engineer and he moved into SRE.

So this is sort of like the side channel into SRE within Google.

So we thought we'd address this because season 3 is all about doing software engineering inside of SRE, so we felt that it was an appropriate bridge episode.

So given that, we also have a new co-host.

I have here, Salim Virji.

And with that, I think we'll just get started.

Healfdene, do you want to introduce yourself before we get going?

HEALFDENE: Thank you.

Yeah, so I'm Healf Goguen and I've been at Google for about 18 years.

And as you said, I've worked in a bunch of software development and just recently started working in SRE.

And I'm excited to talk. STEVE: Cool.

Welcome.

And Salim, why don't you introduce yourself, as well?

SALIM: Thanks, Steve.

It's a pleasure to be here today with you both.

And I'm Salim.

I'm part of Google's SRE education team.

And I've been part of SRE in one form or another for about 20 years.

And I've seen a lot of interesting software development take place in SRE and I'm excited that this season will be digging more deeply into that.

STEVE: Great.

So let's just start from where we begin, right?

What brought you from a pure software engineering role inside of Google into SRE?

Like, what attracted you?

And how did you make the jump, Healf?

HEALFDENE: So I think I've always had a broad range of interests.

I like to focus on something and become a specialist in one area.

But then after some amount of time of going into depth in something, I get impatient and like to move on to something else.

And so I think I've iterated through a bunch of different areas.

And as I iterated through those areas over the past 30 years of my career, I think I slowly realized some things that were more attractive to me and some things that were less attractive to me.

And yeah, we can talk more in depth about what projects led to that viewpoint.

But I think at some point, I realized that the thing that was most attractive to me was-- or most important to me-- was having a service in production and sort of defending the integrity of that service in production.

And so it seemed natural to take an approach, to look at what that meant from the side of SRE as opposed to the side of development.

STEVE: So given that, I've been in SRE for too long, what does it mean to not be running something in production?

What is the opposite of that you felt drawn to in production part of it?

HEALFDENE: In my earliest days, when I was still a teenager and in college, I was just interested in coding, basically, the practice of coding.

So I worked on a C interpreter at Bell Labs.

And we had a few customers.

But it was mostly just the joy of hacking on a piece of software.

And I guess my father was more in theoretical work as an academic.

And he said, you're going to get bored with this practice stuff.

Why don't you try theory?

And so I did that.

I went to Edinburgh and did a PhD in a very theoretical area of computer science called-- well, even maybe it's not as more logic than it is computer science called type theory.

And what I found was that it was fascinating intellectually, and even like I was compulsive about it.

But I didn't end up caring about the results.

And so I went back to working on practice.

And just the hacking clearly was not enough.

So I needed some kind of intellectual stimulation, as well, like some architectural principles to apply to it, as well.

The theory, as I said, was compelling intellectually.

But I felt divorced from the actual practice and having users and caring about a service.

I went to AT&T for a while and we built a voice over IP service, which was actually great.

It was a national voice over IP service.

But it was a much smaller service than something we would build it at Google.

And I enjoyed that.

So it was great to be associated with something that was really-- you know people were using it.

You could see that people were getting value from it.

STEVE: Yeah, like just add users.

It makes it a lot more interesting, I guess.

HEALFDENE: Yes, absolutely.

STEVE: Cool.

So now that you're in SRE, and you've been in here for more than a little bit, what do you find has transferred well from your very distinguished career in software engineering?

What bits of it work?

What has pulled over, like everything, or very few things, or somewhere in between?

HEALFDENE: I think I am still answering that question.

I've been in my role for three months.

So I think I have a lot more to learn about the answer to that question.

So I think I can give pieces of the answer.

But I think there's a lot more to learn about what the answer to that is.

I mean, I think the other piece that we didn't talk about is that you also, as you evolve in your career, you may pick up more of a leadership role.

And so I also found that I really cared about the development of the people on my teams.

And so that piece, I think, transfers everywhere.

So that piece clearly, I still need to-- you know, the role, I think you need the respect of the people that you're working with.

But at the same time, you can apply your experiences to what they're going through and help people develop new skills in those positions.

So clearly, that is an important piece of what I was brought to the organization to do.

But I think if we're talking more in terms of technical skills, it's more maybe that you happen to acquire some SRE skills in the development side.

Maybe I could take that as a hypothesis.

Like as you become-- definitely as you become tech lead, you start caring about your products' behavior in production.

And you need people that you trust to be applying good principles to make sure that the service is running well.

And you come to learn what the things are that do that.

STEVE: Yeah, it can be a little chicken and egg.

Like which came first, your SRE interest, or your actual entrance, your interest or your entrance?

I don't know.

So before you were in SRE at Google, you worked on some other stuff inside of Google.

There's one system in particular that has the shortest code name ever.

I was wondering if you could tell us about that.

HEALFDENE: So that was D. And--

STEVE: The letter D, that's it.

HEALFDENE: It's the letter D, yes.

So that is the network attached server that serves all network attached disk storage.

So it serves all local storage from disks to the rest of the Google network.

STEVE: So given your familiarity with working on D-- I believe you were the tech lead within the D team for a while.

Did you, then, just move straight on to D SRE, or did you find that things would be transferable to other teams?

Or how do you see that experience transferring within different roles?

HEALFDENE: No, I didn't go direct.

So I was tech lead of D for seven years.

And we definitely worked-- like as the tech lead of definitely one of these low level infrastructure services, I would say you're closely paired-- you have an SRE team that supports the product, as well.

And so as the dev tech lead, I think you have a strong view into the overall service.

So I think at some point, I had clear objectives on what I wanted to achieve with D. And I think we achieved a lot of those objectives.

And it became clearer that I could go to a different position right to-- as I said, I was kind of impatient to move on to new projects after I've learned a certain set of skills.

So I did move to a different project.

STEVE: I guess I'm just asking, did you find that being an expert in one particular domain, specifically within Google, this is like this D thing, this disk service, did you find that pigeonholed you when you moved into SRE?

Or did that not really matter?

Like did you find that your skills were transferable to potentially lots of different things?
HEALFDENE: I see.

So, right.

I guess the specific thing is I did move to a different product, but it was a similar scoped product to D. So it's fundamental to Google.

But it doesn't interact with a lot of the higher level services that Google provides, like GCE or GCS or even YouTube.

It's the basis for those things.

But it doesn't have a lot of direct interaction with the higher level components that build those systems.

So in some sense, the technical skills, I guess the architecture specific things of those services doesn't apply broadly.

But I guess the techniques that you learn about isolation and capacity planning and all of those things, those definitely are relevant.

SALIM: Just a couple of phrases that are a delight to the ear-- isolation and capacity planning.

And Healf, which SRE concepts or practices-- I know that a lot of your learning and a lot of SRE generally happens on the job.

Are there SRE concepts or practices that you would recommend that a student pursuing a CS degree learn about?

Because there isn't formal SRE at the university level.

HEALFDENE: I think that's a great question.

I wonder.

Like, I haven't been that close to university or high school teaching, so I don't know whether people are focusing on these things at all.

Definitely, from the software engineering perspective, it always felt like people really weren't taught what the practices of code reviews and unit testing and all of that stuff.

There wasn't a focus on that, right?

It was more a focus on how do you write an algorithm that solves this problem or what is complexity or those kinds of things, right?

And I saw less focus on the engineering practice of it.

And I suspect that it's mostly similar for SRE practices as well, right?

Like, where would you learn about capacity planning in a university?

SALIM: Yeah.

And I know that from my experience in SRE, everything I understand about capacity planning has happened while I've been doing actual capacity planning or faced with issues on a production service.

STEVE: Totally.

Yep.

SALIM: What kind of space or university type curriculum would you envision for SRE practices?

That is, if there were a course on SRE, what are some of the top things you'd expect to see in that course?

HEALFDENE: Yeah, that's a great question.

I mean, I think things we've just mentioned are certainly important in that, right?

So yes, absolutely understanding isolation.

And that can be across lots of-- you can certainly specialize across lots of dimensions in that.

So algorithms for network isolation and CPU isolation and disk isolation.

Like, they're all interesting and sufficiently different that I think you could get a lot out of studying all of those things.

Yeah.

I think thinking about what capacity means, it does feel-- I guess the place I have been is in interviews.

So I do interviews for Google, as many people do.

And I think over the last 10 years, questions that you might have asked 10 years ago that were very difficult for people-- I think the fact that we've been moving to cloud has given people more familiarity with things that we took for granted inside Google but you didn't really see outside.

So I suspect that there are resources now.

So I don't think Kafka existed, or at least nobody knew about it when I started asking those kinds of questions, right?

But then people would just say, well, I'll use Kafka.

Right?

And then you're like, OK, well, now how do I make it more specific?

So yeah, I do feel like, probably, external practice is catching up with things that we did inside the company, or passing.

STEVE: I wonder about that specific topic.

Like, when it comes to something like Kafka or Pub/Sub, it's this weird piece of infrastructure that if you didn't have all the stuff around it, you wouldn't need it, right?

It's like it serves a purpose to connect other things, potentially.

But I wonder, is that enough to know about the existence of that product and the knobs that are on the surface of it?

Or is it fundamentally important for SRE, specifically potential listeners of this podcast who are working in SRE on cloud somewhere, not necessarily inside of Google, to know the guts of that system?

Like, what's going on inside of a distributed message passing system?

Are the products so mature now that the abstraction is good enough, or is it important to know how it all works inside of it?

HEALFDENE: I suspect you're both SREs as well so we can have a conversation here.

But I suspect.

Of course, you can't just take it out of the box and it's going to work, right?

So I think if we were talking in an academic context, then you might imagine that a professor could, for example, have a bunch of test workloads that exhibit different behaviors and say, well, propose a configuration that's good for maximizing throughput or minimizing latency for the following use cases or whatever.

And you could ask people to understand the system well enough that they can tailor it to those kinds of behaviors.

STEVE: I think that's a great way to put it.

I actually just had an interesting dinner table conversation with some teenagers the other day where I talked about how there's many different ways to sort a set of things.

And they had no idea what I was talking about.

And so I talked about like, well, what if it's already sorted?

What if it's totally backwards sorted?

What if it's random?

Like, the different methods you take result in different outcomes.

So I think this helps elucidate what I was getting at, which is, if you want to know the outcomes of running a system under many loads like in production, like you were mentioning, against some weird set of users doing funny things to your system, it is sometimes important to have that mindset of, well, what is the O of n?

What is the sigma?

What are all these?

You think about these abstract computer science ideas of what's the best thing that this can do, what's the worst thing this can do.

Like, are we just talking about the median all the time?

And really understanding its performance characteristics at a more conceptual level besides, just like you said, just taking it out of the box and plugging it in.

So given that, I guess what I'm trying to get at is, do you think that a computer science education is still relevant in SREs today?

Like, my consideration is, yeah, probably.

And maybe we're a biased bunch.

But have you seen cases where folks have no computer science education whatsoever and it's completely fine?

You don't have to have a straight answer either.

This might just be a ball.

HEALFDENE: I mean, I think I'm going to say a computer science education has to be maybe not essential, but certainly extremely useful.

I can imagine somebody coming with a mathematical background maybe.

I think I prefer more engineering background, though, because I think you might try to boil the problems down too much and abstract away from things that are important if you were more focused on the mathematics instead of the engineering of it.

STEVE: Right.

And I don't want to dissuade people who don't have a computer science background.

I'm just asking, do you think it's still additive?

Like, do you think it still helps when it comes to SRE?

HEALFDENE: Absolutely.

Yes.

SALIM: Based on the conversation that we're just having, it sounds like you would assert that it is critical for SREs to be involved as service owners.

That is, rather than encoding SRE best practices in a platform or in a service, SREs also need to come along with that service and ensure that things continue to work.

And those things would be capacity planning, understanding new emerging use cases and so on.

HEALFDENE: I mean, I think one piece we haven't talked about is that at least at Google, there are two broad classes of SRE, right?

So there's one that's more, I guess, systems focused and there's one that's more software engineering focused.

And so clearly, for the software engineering focus, I think having a computer science background is completely essential.

I believe that with the more systems focus, it's still going to be extremely useful.

But I'm not sure that it's absolutely critical.

STEVE: OK.

We have a few questions that we try to ask all of our guests just to be a little consistent, have some horizontal questions.

So you can always pass if it's meaningless.

But what is one impactful change you've made to a team or that you've been a part of?

What is one story that you have around impact that you're particularly either proud of or just sticks in your head in some way?

HEALFDENE: So I think that's the project that I just came off.

So after I finished being tech lead of D I moved to be the tech lead of Chubby.

Chubby is a distributed locking service.

So basically, in a data center, you have many machines that are all trying to do some amount of work.

And you want to guarantee that you can divide the work up, but that when you've divided that work up, only one task is doing a certain piece so that you don't get conflicts where they're trying to do the same thing twice and potentially corrupt data.

And so that's the basic service that Chubby provides.

And Chubby was built in Google 20 years ago, basically, at this point.

And Google was a very different company than it is today.

So really, it had a few enormous services and a bunch of other stuff.

But the core focus of the company was web search, Gmail, at some point, YouTube-- those very large services.

And those large services have the ability that they can run all over the place because they have such a big footprint that it's fine for them to run in many different places.

And so if you have a failure in one location, then you can just basically drain that location and say, please don't send any more traffic there.

We'll send it to the rest of the world.

And that service remains perfectly.

It can continue to serve without any real consequences.

Of course, that's the idealized state, but we got pretty close to that.

As Google started to move towards cloud, really, cloud customers don't want to build that.

They can't build that huge footprint, right?

Like, if you're a relatively small company or even a relatively large company, I guess, it's not going to have the workloads that Google has.

And so they want to reduce their costs by remaining in basically one location, or potentially two locations.

But that architecture doesn't work for them.

And so basically, what we did while I was tech lead was to change the architecture of Chubby so that it meets that need of supporting regionalized services and the reliability model that

they need, instead of the old reliability model that worked for the old Google.

And there are many components to having done that.

And it was extremely interesting-- almost as a marketing exercise, but also as, of course, a deep technical enterprise.

STEVE: That's cool.

That's a deep cut right there.

That's understanding the complexities of the failure domains on the customer's behalf and making sure that the infrastructure here allows for that. That's pretty cool.

SALIM: Yeah.

I'm curious also-- and this is a shift from what we were just discussing.

I'm still really curious about-- as younger audiences or younger people come to the practice of software engineering and site reliability engineering, what kind of guidance would you give them so that they can improve the internet as we have it today?

And so I guess there are two ways that we can look at this.

What would you tell teenagers or people who are embarking on a career in internet services?

And also, what can we generally improve about the internet?

HEALFDENE: Yeah.

I'm sure there are many dimensions to that question.

But the ones that I feel most directly reflect on my experience is that, I guess, it could look from the outside the internet is done, maybe.

Maybe you have that perception.

But inside Google, it's very clear that things are constantly changing, right?

And so there remains a huge amount of work to do and very interesting things to think about.

And there's clear impact to what's happening there.

So if you're excited about AI and machine learning, then Google is doing amazing things in order to support that in a better way.

And it's really interesting across the full band of potential interests, right?

So from power infrastructure all the way up to what are the models that support those kinds of things and the more abstract approach to that, right?

SALIM: Sure.

Power infrastructure is a really hot thing.

No pun intended.

It's very much on the minds of, I think, a lot of people, especially the younger folks who are coming into a world where there is a very different landscape for power consumption.

And I think also as an awareness of how power intensive a lot of our deceptively small computers can be because we've got these devices that we can hold in our hand.

And yet behind these smartphones and similar devices, there's a tremendous amount of computing power in the cloud.

HEALFDENE: Yes.

So I guess one of the projects-- it's a long time ago at this point.

But I did make a relatively simple change over a few weeks in Bigtable, which I had worked on before D.

And it saved some substantial amount of power, which did feel like that was a real accomplishment, just to be able to make relatively simple software changes that actually change how much money Google is spending and how much power we're consuming.

And it's definitely possible to have that kind of impact, both from the development and from the SRE side, right?

STEVE: Yeah.

One of my favorite ISMs that I've learned working at Google is that a small percentage of a large number is still a large number.

So when you can make a little bit of an improvement to a pretty large system, you can have pretty dramatic impacts.

That can work in both directions, I suppose.

But when you're talking about cost or energy savings at a global scale, pretty soon, you're talking about real numbers.

HEALFDENE: The alternative view, unfortunately, though, is that as you make things more efficient, it just means people use them more, right?

STEVE: Yeah.

SALIM: Yeah. Induced demand.

HEALFDENE: Yes, exactly.

STEVE: So big picture, get your futurism hat on.

Salim was alluding to it before.

Like, is it good that society uses these computers so much?

Are they reliable enough?

Can we actually depend on them, and should we depend on them more in the future?

HEALFDENE: I think I'm going to beg off on the question of AI in particular.

But if we talk about moving to the cloud, it feels inevitable, right?

Like, there are substantial efficiencies, I think, for many companies.

And so they're just going to do it regardless of what we think.

And it should lead to more-- again, with the caveat that I just gave that maybe they'll do more because it's more efficient for them.

But it is more efficient for them.

And so in terms of reliability, I guess we can't guarantee how any individual company is going to do in terms of continuing to honor the commitment we have.

But I can definitely say that we all have a strong commitment to the reliability of our systems.

And there's a huge amount of investment in making sure that these things are reliable.

STEVE: Great.

Well, that was wonderful.

Thank you for your time.

This has been an exciting discussion.

This is a great way to help us kick off our new season of the Prodcast.

So I really appreciate your time.

Salim, how about you.

Any quips?

SALIM: I don't have any quips to share.

STEVE: All right.

Worth a shot.

SALIM: I mean, I can lean back on the old, hope is not a strategy.

And I think for software engineers and folks with CS background, it's very clear that we can approach engineering problems methodically and with a lot of discipline.

But my time in SRE is also a reminder that no matter how much we plan, our systems will always find a new, chaotic behavior to throw at us. STEVE: Yeah.

I believe it.

All right.

Well, Healf and Salim, thank you both today.

This was a great talk.

And as always, may the pager stay silent and the queries flow.

HEALFDENE: Thank you so much.

SALIM: Thank you.

VOICEOVER: 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 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.