EPISODE 1359

[INTRODUCTION]

[00:00:00] YB: The last 15 years has seen the emergence of cloud-based developer API's and services as dominant components of the developer tool chain. As a result, there has never been more power at developers' fingertips. But making that power usable and accessible is a challenge that is shared between the providers and the consumers of these services. For the past 15 years, Google's Developer Relations team has been bridging the gap between Google's internal engineering teams and the broader developer community that consumes the API's and services.

My name is Yaniv Bernstein, your guest host for today. And I am excited to speak with Luke Mahe, a longtime engineering manager at Google DevRel, about the history of Google's developer offering and the critical role DevRel plays in shaping and promoting that offering.

[INTERVIEW]

[00:00:50] YB: Luke, welcome to Software Engineering Daily.

[00:00:53] LM: Thank you for having me.

[00:00:55] YB: So just before we jump in and talk a bit more about developer relations at Google, please tell the audience a little bit about yourself.

[00:01:03] LM: That's a big question. I wish I had lots to say. But I've been at Google going on almost 15 years now. So I spent pretty much most of my professional life working here. I started in 2007 in Mountain View. Did a couple of years there, then moved to Sydney, and been here ever since.

[00:01:23] YB: If I understand correctly, you spent that entire time within the developer relations organization at Google. So you've really had a front row seat to nearly the entire history of that. So tell me, from your point of view, what is developer relations at Google all about?

[00:01:38] LM: Yeah, it's interesting, right? DevRel, as we'd like to call it, is an interesting mix. I often like to coin the term we're kind of software engineers with personalities. We are kind of like the face of the product, but from the engineering point of view. So we work on developer products. So nothing that kind of really faces the commercial world. So we wouldn't say work on. I'm trying to think of anything as a non-developer.

Actually, **[inaudible 00:02:05]** has an API, right? So if there's the ability for somebody else to build on top of our platform, usually there're developer relations that are associated with it. And our job is to engage with the external developer community. So depending on where that product is in its life cycle, if it's just starting out or if it's a bit more mature, we'll have different opportunities of doing different things. But the first bit is to just understand the product, understand our users, understand what they're trying to build with the product and then seeing if we can enable them to be successful.

Actually, he's my boss, but he came up with a good term a couple years ago called the zeroth customer. So our job is to be the zeroth customer. So we want to be the first people there building on top of our platform, figuring out what it can do, figure out what it can't do, figuring out where it sucks, going back to the engineering team, working with them to make it better. So when it comes to being used by the external community, they don't face those problems.

And then also on top of that, listening to their feedback. So understanding what they're trying to achieve. And if they can't do it, go back to the product team and being like, "Hey, this person has a cool idea of being able to do this with our product. Can we make it happen?"

[00:03:24] YB: So in a sense, you see ourselves as the conduit between the internal teams building developer products and the broad world of developers out there at Google. I wonder, in terms of the way where DevRel sits within the broader Google business, do you

consider it a marketing function? How do you work with some of the other organizations such as pre-sales or sales engineering? Where does DevRel fit into that system?

[00:03:52] LM: So I know that other organizations that have developer relations, their DevRel team do sit under marketing and it is a marketing function often used to do, as you said, pre-sales, or post-sales, or drive growth. At Google, developer relations has always been a purely engineering effort. So we don't really fit in with marketing as well. We'll sit beside them. But our goal, first and foremost, has always been building the best product that we can listening to the users and bringing feedback to the product to improve that and less about driving growth or driving adoption and driving sales.

And we're not like a PSO style organization where we will not go and come along after you've bought a license and then do the integration for you and that sort of stuff. But we must prefer to work on like the open source world of we'll build examples, or client libraries, or documentation, or we'll do videos, or we'll go talk to conferences to see if we can scale that up rather than just being a one-to-one sort of connection between Google and a specific customer.

[00:05:06] YB: So that makes sense. I think you've kept it quite pure in Google DevRel. But it does raise the question, I suppose, of how do you then measure the success of developer relations if you're not a marketing or a sales function?

[00:05:17] LM: Oh, that is the question everybody always asks, and it's very, very hard, right? You can always tell when you've done your job poorly, but it's very hard to tell when you've done your job successfully. I feel like with developer relations, if we do a good job, often, people don't know we exist, right? If we can say there's a new product launching and we're going to go to market on the API, understanding use cases of what people hoping to build with these APIs. So say it was Android. Before Android even existed, developer relations main focus was basically education. Telling people android existed. Building sample apps and sample products and examples out there of common problems people are likely to solve, right? A lot of apps out there are navigation and a list view. And you tap on the list view, you see a detail bit and then it connects to some sort of cloud database and get some data, right? Maybe we should build an example application out there that is like

the best version that this sort of thing can be. We'll will open source and anyone can take it and fork it and then use that to build their own products off that.

Otherwise, I'm trying to think of some other ones. Say, like, Maps API was a little bit more mature in its time. So that was more understanding, "Okay, what are businesses trying to do on it? What is the intent? And where can we go?" So what are people really trying to build?

[00:06:43] YB: Yeah. Okay, that makes sense. So it is quite situational. And you did mention the Google Maps API. So I thought it would be good to go into the history books a little bit and talk about the Google Maps API, because the Maps JavaScript API in its day was something that really exploded and developer relations was there quite early. You had a front row seat to a lot of that. So could you tell us a bit more about those days and what it was like?

[00:07:12] LM: Those are the wild wild west days. I can't claim all credit for like initial Maps API DevRel. Like that goes to – Some of your earlier listeners may know this name, Pamela Fox. She spent a lot of time doing developer relations with the Maps before I kind of took over in her role as she moved into Wave.

But Maps is – Yeah, I've always loved Maps and Maps API, right? Because even how it started is such a cool story. Like we had the Google Maps product and someone reverse engineered the Maps product to mash it up with data and create a new product out of it. I think it was one of those was like the Chicago crime map where they plotted where all the crimes were in Chicago on a map. And there was no API for this. They just reverse engineered the JavaScript and put it all together and threw it up and people could see it, right?

And I always saw the value of Google providing a platform because we're really good at like building platforms, but understanding people's connections to their local communities and things that are really important to them. We don't have that idea. We probably shouldn't. So if we could end up building a platform where anybody can use it to have an impact on their local community, then that for me just like makes me feel good, right? You see people who

do really awesome things of building little maps or little applications that just like support their local environment or their local community and that sort of stuff. And I'm like, "Well, hopefully, the world is becoming slightly better place if we enable these sorts of things."

[00:08:48] YB: Yeah, and thanks for that. I love that story as well. Google was not in the business of developer tooling by and large back then. And as you said, Google Maps was not intended to be a developer tool. But I think it really speaks a lot to Google's culture back then that when you saw this happening and you saw a community of developers building around this product and these incredible possibilities, that Google and the developer relations team really played an instrumental part in enabling that. And in a way, that was the first really big API on the Internet that got the sort of broad community adoption.

[00:09:23] LM: Yeah, exactly, right? Like we've had large API. The Ads API, it's a large API. We've had Android and those sort of ones, newer and quite large. But at the time, I'm pretty – Someone could correct me if I'm wrong. But I think the Maps.js API was – In terms of users on sites was one of the largest used APIs in the world. And this is kind of pre-ajax sort of world, pre-jQquery if you would call it that. That, yeah, it was really cool out there. And that was actually quite a fun, right? Trying to understand from a technical point of view to build an application that we worked on IE6, right? IE6 was one of our biggest browser bases back then. And anybody who's a web developer would understand how challenging it was to support both IE6, Firefox, Safari back then. Life is a lot easier now if you're a web developer.

[00:10:18] YB: Yes. And it's amazing to think all this was less than 15 years ago. And when you see the ubiquity of APIs and the API economy these days, that less than 15 years ago the technology was immature and the idea of a broad community of developers built around cloud APIs, if you want to put it that way, was completely new.

[00:10:39] LM: Yeah, exactly.

[00:10:41] YB: So now moving more back to the present day. I wanted to talk about Google Cloud Platform, or GCP. Despite explosive growth in GCP, it's still very much a challenger

platform relative to AWS and even Azure. And so I was curious to understand, how is Google DevRel helping to get more people using GCP?

[00:11:03] LM: Good question. I mean, I don't focus on GCP. So I'm not going to try to answer the specifics for that product. But I think like with a lot of developed relations, a lot of it is you got to figure out where in the life cycle the product is and then what is the best effort that people could do to get the most ROI. I mean, obviously, pre-Covid, the world was slightly different. We used to go to a lot more conferences and talk. I actually meet with the developers there. These days, things are changing. It's going to be interesting to see what happens next couple of years. But specifically with GCP, I think a lot of the challenges come to understanding what the customers are trying to do. What problems they're facing on the other platforms, right?

AWS is a huge platform. It has its own intricacies, same thing with Azure as well. So it's understanding a little bit what Google could do differently compared to those other products, whether it is performance, whether it is being able to run your application more closer to the user, or whether it's your ability to get costs down, or whether it's just ability to manage resources, or just the tooling or the experience. And then also it goes on to like what is the team and the organization, what's their experience?

A lot of people are using AWS because their whole engineering team came from a company that all used AWS. So they're just using what they used. Some people are GCP. And then it takes a lot of education and outreach and actually talking to people to be like, "Hey, I know you're using AWS and it's this. But GCP can do it here, cheaper, faster, harder, better, stronger," all that stuff.

[00:12:50] YB: Right, and you get that credibility from having actually used the tool in a sense as a zero with customer.

[00:12:56] LM: Yeah, yeah, right. So that's the thing, is it all comes down to really understanding your customer. And it's not about getting a sale or that sort of stuff. It's about what problems are they facing in their business? What are they trying to build? And how can we enable them to be successful on our platform?

I often will say, with DevRel, if telling them to use a different ARPI or a different product is the right solution to them, then we should tell them that, right? But what we should do, and specifically at Google, is our aim is to make the best product that exists, right? And if somebody else is doing it better, we should look at why it's better and how can we change, or develop, or improve to make a better product?

[00:13:44] YB: Yeah, that makes a lot of sense. So you touched briefly on the community aspect of DeVrel and how you used to attend conferences. That's obviously a lot more challenging these days. But what are some of the other ways that you engage directly with the wider developer community? And if you, as a developer out there, wants to engage with DeVrel, how can they find you?

[00:14:05] LM: Right. Yeah. So there're a couple of options that we used to take, right? There would be going to conferences and giving talks. These are usually where you had something new to announce or you're trying to just basically bring awareness or shift or just tell people that you existed. And then on top of that we'll do things where if you're releasing a new feature in the API, maybe you'll do sample code, you'll do a blog post if it's big enough. You'll update the documentation. We will write client libraries. Maybe there's some general problem that people solve by combining a couple of different APIs. Maybe it makes sense to have a client library, which is a single layer in front of that where a user can just interact with that and it abstracts away the complexities behind it.

And then we also do things like YouTube videos as well, which is for more content, which is a little bit less fresh and new and more something that might stand a little bit more the test of time. There's nothing worse than seeing a YouTube video that's like three months old that's already out of date, right?

[00:15:12] YB: Absolutely. So yeah, one of the things, a big challenge of being a developer these days is simply the huge variety of choices out there. And after having had my hands off the tools for a few years, I started getting back into building some web applications myself in the last couple of months. And it was really quite overwhelming just the sheer variety of choice out there. So if I'm developing an app, do I use pure native Android, or

React Native, or Flutter, or progressive web apps? For the backend, should I use Firebase, GCP, AWS and so on and so on? So to what extent do you see the job of Google DevRel to help developers navigate the huge amount of choice that's out there?

[00:15:57] LM: Yeah. I mean, the answer to your question is yes to all, right? There is so much choice out there that it is really, really hard to understand. I think like when I talk to startups and other businesses out there that are building, my general feedback is understanding what is the skill set of the team, right? The most important thing to do is to be able to build the product. So depending on like the resources you have at hand, there's no point if you have a bunch of Java engineers who have never done any frontend stuff to go maybe ask them to build a React app, right? When maybe it would make sense to be able to use a different tool chain that would make sense for them.

Google's responsibility comes down to educating them on like what is available? Again, it's like what are the problems you are trying to solve? What resources do you have at hand in terms of the skill set of your team, your budget? Are you a small company that's outsourcing a lot of this? Do you have a couple of engineers? Do you just have a smart idea and you just want to get it out there to kind of either get awareness or start raising adoption or funds, all that sort of stuff?

I think it often comes back to what are you trying to do? Where are you in this sort of life cycle of your development? Are you just starting out? Are you well along? And then looking at what exists. Yeah, I mean, anybody who's a web developer at the moment knows that the tooling and the product of the day changes every six months, right? There was Angular, there was React, there was jQuery, there was Vue. There's Next.js now. There'll be next Next.js soon enough most likely. And it's the same with like Android and even iOS.

I mean, one of the frustrating things I often found was I would do some iOS development or some Android development, and I'd come back and look at it in six, eight months and it wouldn't build anymore, right? I just get update and updated the libraries and then I'm like, "Oh, it doesn't build." That shouldn't happen, right? We're like 20 plus now. We shouldn't have these things that's not working. We should have the tool chains. We should have

everything already sorted out so you can upgrade code from like six months ago, or 12 months ago, or two years ago and it just works.

[00:18:30] YB: I definitely agree with that. I guess related to that, how partisan are you? And what I mean by that is when you think about how to talk people through the developer experience, do you include non-Google alternatives and non-Google sponsored alternatives? Or is your advice mostly for people who are committed to working on a Google stack?

[00:18:51] LM: My personal opinion, and it probably is slightly different to others, but I am not partisan. I feel you should use the best tools to solve your problem most efficiently. And I think it is Google's responsibility to be the best tooling. And if we are not, we need to work on that. So if I've spoken to people before and they're like, "Oh, I want to do this cloud thing, but we're only in this region and there's no GCP in this region." I'm like, "Okay. Well, use AWS," right? That's the right thing for you. That's the right thing for your users. And that's right thing for your customers. Do that, right?

At the end of the day, it's more important that the business succeeds and we end up with really well-built, well-working applications that exist in the community and the ecosystem than somebody making a poorly written application just because they want to use a specific technology.

[00:19:49] YB: Absolutely. And so I guess on that note, what are some recent and upcoming technologies Google or otherwise that you are particularly excited about?

[00:19:59] LM: Oh, I don't know. I'm doing a lot of web stuff recently and I'm quite excited about LitElement, which is kind of like a web sort of framework based on Custom Elements, which is similar-ish to React, but it doesn't come with a lot of the React bloat that normally comes with the React application. So what I like about LitElement is the fact that they're kind of encapsulated elements that can be used anywhere. And they're based off like proper browser spec of custom elements.

I wonder what else. I mean, I'm really excited about really interesting things in the ML space even though I don't understand the ML space at all. I really want to learn. But I just haven't got the opportunity yet.

[00:20:49] YB: Actually, it's funny you should mention ML and AI, because I think one of the big challenges with ML and AI is that there is a lot of excitement around it, and of course a lot of genuine utility. But the big challenge for developers is really understanding what's going on. So I don't know if you have an opinion on this, but I'm curious to get your thoughts on what role developer relations can play in actually helping to close that knowledge gap and that understanding gap? Because I know that AI is something that GCP is positioning itself as being particularly strong at. So how do you kind of close that gap?

[00:21:29] LM: Yeah. I mean, this all comes down to education, right? When I've done any sort of ML stuff, I don't really understand it enough. I'm just saying, "Oh, you've got a graph that kind of curves at the bottom and you want less curviness at the bottom. And if your results start going back up the other curve, you've done it wrong." So I don't understand enough about like maths and the science about it. But what I like is when people build and create models and then have nice APIs on the final field where I can say, "Here is a set of data. Here's a request." And then the result comes back and I'm like, "Oh, cool! You've done something smart, smarter than I can," even whether it was like recommendations or understanding and spotting interesting things. Like I've got one of my old teammates that's been doing some interesting stuff where they're using machine learning to detect litter floating in the ocean so they can kind of, from satellites, understand like where there's a whole bunch of like rubbish floating in the sea. And I'm like, "That's really cool. I wish I knew how to do that."

[00:22:31] YB: Yeah. And the applications are incredible. So maybe let me frame the question in a slightly different way. If the person listening to this podcast is a typical career developer, loves technology, loves keeping up with what's cutting edge and works at a relatively standard tech employer, what should that sort of person's way of engaging with AI and ML be?

[00:22:59] LM: Yeah, interesting. I mean, there's a definitely a couple of different avenues, right? There's a bunch of really smart people out there making interesting YouTube videos where they will explain a lot of like the basics of AI and ML. Then we've Google's products, which is like Tensorflow, where people can go read the documentation. And it's got really great examples, and data sets, and models that you can download and run locally and experiment with. And at least like for the way that I would learn, personally, I've always been the sort of person who learns just by messing around with something. So I would enter the space like that, is I just start reading those documentations and playing with the examples and just like putting in a bunch of input and just seeing what the output is and seeing if it makes sense to me. And then changing the input and then seeing how the output has changed.

[00:23:50] YB: Yeah. It occurs to me that one of the challenging things about being a developer these days is very hyped and very promising technologies that are actually very complex and difficult to get your head around. And we've been talking about machine learning, of course, crypto and blockchain is perhaps even more the case. So, yeah, I feel like one of the biggest challenges right now for a developer is to understand what tools and what technologies they should really be investing in and what level of depth of understanding they need to have in order to make a meaningful difference to their abilities and to their career as well.

[00:24:27] LM: Yeah. And there're both like – Often, people take the thing where if their hobby was their day job, they'd hate it. Where like I think a lot of people who your day job may be the software engineering part. But if you start joining an open source project or finding other sort of hobbies of using machine learning to this. I've got a friend of mine who's trying to hook up his tesla battery with like the local weather system to be able to determine whether it's going to be sunny tomorrow or not. And then that'll readjust using ML how we should use his battery during the day. And it's a hobby for him, right? But I think if that was his normal day job, he probably wouldn't either find the time to do that or really want to do that. Often, if you find it more interesting and it's not your job, you'll actually end up doing something cool about it.

[00:25:19] YB: Yeah, that makes sense. I guess going to the other end of the spectrum, we talked about some particularly challenging technologies to use. There's also a lot of excitement at the moment about no code tools. So I guess that starts to challenge the concept of what does it mean to be a software developer? And I'm keen to get your take on whether no-code tools really do have the – The modern generation of no-code tools have the ability to democratize software creation in a way that hasn't existed before. Or is this one of these old stories where no-code has been a dream for the last four decades or longer?

[00:25:59] LM: Yeah, I mean that's been a dream for as long as I've ever been a software engineer. I mean, all these things conceptually are just layers built on top of other layers of abstraction, right? So the no-code stuff is often a nice little GUI that under the hood has some APIs, which will then construct something that can be interpreted later on, that depending on what the output I, may or may not output native code or eventually go through some sort of interpreter.

I mean, this has been – Like not the no-code, but even just like the write once run anywhere sort of dream that was around for a while especially when it came to mobile, because nobody wanted to do Java and Objective C at the same time, because you put your square brackets where your round bracket should be and nothing would happen.

So I think it's interesting. I don't know if anyone will ever build just like a full-scale compelling app at the moment with no-code. But if you are a non-tech person who has a really interesting idea and you just want to try and get it out there to kind of prove some sort of viability, then use whatever tools you can, right? At the end of the day, you've got a problem that you want to solve and you think you can solve it. Who really cares? Solve it as how best you can and then make it better later on. Because it's better to get something than nothing.

[00:27:25] YB: Yes, I agree with that. And, of course, everything is an abstraction sitting on top of an abstraction and so on. And then it's all just electrons running through silicon. The really interesting question for me is we've had all these layers of abstraction and yet the core skills of a developer and what makes someone a software developer haven't really

changed from the days of writing assembler. And I think no-code offers the promise of democratizing. And the real question is, is it simply another layer of abstraction or does it have the opportunity to create a new class of people who can create software?

[00:28:02] LM: Yeah. I mean, I hope, right? I think that would be the dream is it would be great if everybody had the ability to create software, because there are often – People have got really cool ideas that just don't have the ability to get them out there. But even under the hood, you're going to have software engineers who are going to be building the no-coding platforms, right? Unless you can build a no-code platform with a no-code platform, and then it becomes self-aware.

But I think the real job of software engineers is less about writing code and it's more about simplifying complexity and abstracting the difficulties away to produce a simplified layer. And this is why I think APIs are really good, right? APIs produce a very simple to understand interface for someone to talk to you. But under the hood, it does a lot more work. But they don't really need to know that, right? They can ask a request. You say , “I’m going to give you the response back exactly like this.” How I get the response, don't worry about that.

And I think a lot of people can understand that sort of connection between the code you're writing and the backend and not needing to understand the backend and just being like, “Whoever wrote the backend and the underlying stuff knew what they were doing.” And I feel like that is kind of the future of just more APIs.

[00:29:24] YB: I think that's well said, the task of managing complexity. So, DevRel folks always seem to have a lot of fun. As you put it, developer relations people are software engineers with personalities. Some of them, I’m told, even have good personalities. So is developer relations in your view a distinct career path? And if so, for folks who may be interested in following that path, what is your advice?

[00:29:48] LM: Yeah. I totally think it is. It's definitely a career path that has been growing. I think when Google first started developed relations, we were one of the first people specifically having it as an engineering arm. We're the first people to kind of do that. But

now there're developer relations for all sorts of companies out there. Basically anybody who has any sort of developer offering will usually have developer relations going with it. And if you like talking to people, if you like engaging with others and educating and teaching, but also building and creating, it's a really good career opportunity for a lot of people. It kind of mixes the world of just writing pure code as well as being able to kind of go out there and enable other people to build successful stuff on top of what you've done and enabled other people to do. It kind of feels good, right? If you see interesting startups out there and you're like, "Hey, I help them do that."

[00:30:47] YB: So if I was listening to this and I wanted to maximize my chances of landing a job in developer relations, what sort of things should I be doing?

[00:30:56] LM: Right. So there're things like community groups. Like Google has one called GDGs, where you can go, and they're called Google Developer Groups. And they have meetups and chapters all around the world. You can go then you can meet other people who are working on Google stuff. As well, Google – Well, this is obviously before Covid. It's slightly changed at the moment. But Google would often send Google engineers to go and speak at these events and talk about new products that Google are doing or interesting things. But it's a great place to go and meet other sort of like-minded individuals who are also in the same space as you.

And then on top of that, it's like being more involved in open source world, right? Like a lot of developer relations people come from the open source world because they understand what it's like to kind of work in the community. Understand what it's like to kind of build things out there where anybody can sort of look at what you're doing and trying to like get adoption and get people to kind of use whatever you're building. And then on top of that, it's more just like fixing, right? You see projects that you might be using on GitHub and then someone's got a bug or something there and you just send a pull request and fix it. Often, people can be like, "Oh, you keep contributing to our React app. Would you like to come and work for us?"

[00:32:09] YB: Yeah. Bottom line, just really get involved in the same communities that the developer relations people are in and show that you've got what it takes to be one of them in a sense.

[00:32:20] LM: Yeah, great.

[00:32:21] YB: Finally, Luke, I was just curious if you had any big predictions for the future of the developer experience.

[00:32:30] LM: Interesting. I mean, I definitely feel like APIs is the future for a lot of things, right? We're going to be moving to a world where things will be acting on your behalf. You'll have devices. You may have other things. So being able to understand sort of like a trust model through an API and kind of seeing where that takes you is going to be an interesting place. I really believe that like the future of us is like basically having a device now in our pocket and we just walk in that sort of like does a lot of like the communications forest between each other or going forwards. But I don't know. It's interesting. I feel like machine learning, AI and then connecting individuals is kind of like the future space, especially with the world that we're moving into now where it's a bit more remote and figuring out how to communicate and that sort of stuff. I feel like that's probably where we're going to be ending up.

[00:33:32] YB: Really interesting. Luke Mahe, thanks very much for a fascinating conversation.

[00:33:37] LM: No worries. Thank you very much.

[END]