EPISODE 1466

[INTRODUCTION]

[00:00:01] JM: Running a database company requires expertise in both technical and managerial skills. There are deeply technical engineering questions around query paths, scalability and distributed systems. And there are complex managerial questions around developer productivity and task allocation.

Sam Lambert is the CEO of PlanetScale, which is building a modern relational database infrastructure. Before PlanetScale, he spent several years on infrastructure at GitHub. He joins the show to talk about his work at PlanetScale and the vision for the company.

If you're interested in sponsoring Software Engineering Daily, we reach over 250,000 engineers monthly. You can send an email to sponsor@softwareengineeringdaily.com to learn more.

[INTERVIEW]

[00:00:41] JM: Sam, welcome back to the show.

[00:00:43] SL: Thank you. It's awesome to be back.

[00:00:46] JM: You are the CEO of PlanetScale. And I have a lot of high-level questions, some lower-level questions. I want to start by just asking, how does building a database company today compared to how it was in the past, maybe when you think about other database companies in recent memory such as MongoDB, or even further back like Oracle?

[00:01:10] SL: I think it depends what you're trying to do. I mean, you can still go and build a database company if you just focus on the actual database. But I don't think that's going to cut it in the long run. I think you have to have something that's stable, reliable, and does its fundamental job, which we know we have that covered with our kind of backend tech, the tests. But also, I think, in this modern world, and if you look at where the industry is going, it's now more important than ever to build something that's loved by developer. Developers are so important to every part of company life in the tech community. But even at non-tech companies,

developers provide a lot of leverage for the businesses they're a part of. And if they don't love the tool that they're buying or using, it's not going to cut it for the long run. So I think in this modern world, it's about building something that's creative, amazing to use, great developer experience, and is robust and reliable.

[00:02:07] JM: Now, what I would say is, each of those companies had kind of a core innovation. Like, I think MongoDB, just the core innovation, was kind of a usability just in the sense that it was the database that felt like it was JavaScript-oriented, or felt like it was document-oriented, that had the most traction. And similarly, PlanetScale has this core innovation of having a well-designed MySQL scalability. Why wasn't relational database scalability solved by the time PlanetScale came to market?

[00:02:42] SL: It's a really, really, really good question. And I think Mongo is the absolute right case to think about. So people kind of hate on Mongo. I certainly was not a Mongo fan when they first came out to the market, because the things they were saying, I think, was really not true about databases. And they had a lot of issues. And I remember, I worked for a place that ran one of the larger Mongo clusters at the time, and it was not fun.

However, Mongo, I think is a great lesson for everybody. And that is that developers are not necessarily always going for some kind of database purity, or what some standard, or paper says. They primarily want to be productive with something that's good enough, and that's fast and, and works for them. And I think Mongo was very successful by understanding that and getting in and building something that got out of developers' way. And they did an immense amount of work to make very, very, very good developer experience. And they kind of did the rest later, which is – And they built a fantastic business on top of it.

I think it's like I think Mongo exists today because of Postgres, MySQL, kind of asleep at the wheel a little bit when it came to experience and kind of just focused on the database fundamentals and passed on a lot of the pain. It satisfied that we're solving hard problems for people. So we can pass on some of that pain back to the user. And it took too long for the MySQL world to kind of catch up with that. And I think we're trying really hard to kind of make gains on that and make something that's supremely usable and really, really good.

But now MySQL still dominates. People talk about MySQL being over and dead. And I think that's not true. Still, most of the major websites in the world are still running it, and a massive amount the Internet. I think it's like 79% of the Internet is still run on PHP. You've got to imagine. Would you say it's unfair to say 90% of PHP websites probably use MySQL?

[00:04:34] JM: That's probably right.

[00:04:35] SL: Right. So still a massive amount of the Internet is running on MySQL. And there's something to recognize there. And we're going to carry the baton forward. It's been around for 25 years. Hopefully, it'll be around for another 25 and it'll get a lot better because of companies like us.

[00:04:49] JM: What is percentage of the Internet that's WordPress? Something like 80%, or 70%, or 60%, or something? Do WordPress installations want Vitesse?

[00:04:59] SL: Funnily enough, we have a customer that's a top 3000 website. They run on PlanetScale. And their backend is WordPress and their frontend is Next.js, a common pattern we're seeing now. And I saw there's someone doing this for Drupal as well, which is again, really interesting. It's like companies, ecommerce, media, all of these like large sites were built on WordPress. There's so much data and backend that's all built in WordPress. But they want to have a new kind of frontend and want to leverage some of the amazing tools like Versa, or Netlify, and these new frameworks. And now they're treating WordPress as like this API middle layer and kind of doing headless WordPress. So we have customers that are doing that. And obviously, WordPress runs on MySQL. And when you get to large scale, the fact we have a drop-in replacement for MySQL, we're seeing people migrating off of Amazon RDS onto us to use these kinds of solutions. So I think, actually, that this new architecture for WordPress, and then backend services like ours are probably going to keep WordPress in the game for even longer.

[00:06:02] JM: And just as an example, can you just explain, like, if I'm running some amazingly complicated or high-traffic WordPress deployment, what am I going to see in terms of – If I move my core database to PlanetScale, what am I going to get out of that migration? Is it going to be a performance gain, or is it going to be more around the usability layer at this point?

[00:06:28] SL: It's both. And I think both improving either is very, very impactful for an organization. We see that we kind of round off some of the bursts. So if your traffic is low, and it's the middle of the night, some of those less optimal queries or less optimal patterns are less problematic when your database is underloaded. Problem is when it gets to be very loaded, that's when things start to get really problematic and can cause slowdowns around the entire system.

And having a more mature database stack like ours in the backend kind of takes away some of that pain and pressure and gives a route towards horizontal scalability, which means this kind of like more bursty workloads and workloads that are more complicated and less optimized run much better.

Also, we have features that protect the database a little bit more. So like, if you just truly overload a database server, it just falls down, right? That's very clear. You go from like query slowdown for a little while, and then they just stop and it stops working. The fact that we have scalability, and we have features like hot row protection. So if you're like hammering a row with updates and inserts, we can queue those and send them down to the database in a way that's stops you completely hammering a row.

Same with reads. We can detect if you're doing – If you've sent the same read queries, we can catch them all at once and serve them with a single query. So that kind of intelligence between the app and the database really does help with kind of running apps that are bursty and complex. And one of the big WordPress sites we moved over was on Aurora, which kind of claims to be a real solve for a lot of these types of problems. And it just wasn't they were having really bad reliability. And so they moved over us very quickly. And they've been very happy since.

[00:08:14] JM: I'd love to know more about the comparison matrix when people are looking at hosted relational database solutions. I imagine you have some selling points on each of the comparisons like Aurora. And I don't know what the other MySQL scalability options are. But do you have like a point-by-point comparison or some of the main ways in which PlanetScale is able to Trump its market competitors?

[00:08:46] SL: Yeah, actually, if you go on planetscale.com/enterprise has a comparison table. Primarily, this one compares scalability. And I'll give a brief overview of that. One way we massively differentiate that's obvious is usability, right? And that's obviously very subjective in terms of like one being more usable than the other. I mean, we know that Amazon experience and taste is not a strong suit of theirs. And we find that a couple of our customers, they've got into the position where they had a decision, which was like try and hire database experts. Very hard things to do nowadays. Or move to a platform like ours where they don't need to. Don't have to have DBAs. And a lot of them choose that. Because it surprises me. The money they make on Amazon RDS and how little they do for you is wild for me. Like it's just incredible markup on top of the machines they provisioned for you. And it just does very little.

And we have a deeper feature set, things like branching that we brought out there that we created and it’s showing – It's a whole new way of thinking of staging environments and deploying schema changes. None of this stuff exists in these rival platforms. But then it comes from a pure scalability perspective as well. So we've got customers moving to us right now. They've got 20 terabytes of data, and they had to sync a terabyte every month. That's really, really hard to do on these types of platforms.

So like, if you look at MySQL RDS, the max storage size you can get a 64 terabytes. Well, we have customers with clusters of Vitesse and clusters at 20 petabytes. So a massive amount more storage available on PlanetScale. Same with the amount of cores. We have clusters of 50,000 cores. The most cores you can provision onto a single database with RDS MySQL is 576. And you get only five replicates. Well, how can you run a serious website for five database replicas? It just doesn't happen.

And so there's all these people that start their companies on these products. And then very rapidly, they get to the point where it just stops scaling for them. And they're pretty stuck. And we talked to a bunch of customers that are trying to unwind gigantic architectural balls of twine that they built around Amazon RDS. Customers have built their own sharding layers. They build app logic for sharding. They've split separate tables out onto clusters. And it’s just really messy. And, really, a lot of operational overhead for a product that charges as if they should be solving all these problems for you.

[00:11:15] JM: It's so surprising to me that a product is with as much market penetration as Aurora hasn't solved sharding, doesn't have built in sharding. I guess it just speaks to the level of engineering difficulty that was there for a long time. And I suppose, you kind of needed Kubernetes to build Vitesse. So I guess just the database lineage demanded that things evolved this way.

[00:11:39] SL: Right. And that's actually really – The lineage. You're talking about the lineage of the Vitesse is really interesting. And the reason – And it's such a strong suit for Vitesse as well. Like you see these databases now, they're like, "We're cloud native. We do all this sort of stuff." And it's like, "Yeah, cool. Like you're building from the bottom-up with very few large customers." I mean, their customers are going to run into – You don't want to be the biggest customer of your database platform. So we kind of joke that we'll will never find a bigger customer than YouTube where this thing was built. And it was built on Borg, which was the predecessor and inspiration for Kubernetes.

And so it was built in a world where there was no true persistence, and it was built for failure in a very dynamic, but anti-fragile environment and still running on MySQL, which is very robust. And so it was born in the fire almost of trying to scale a colossal website with not just millions, but billions of users. And that is a real strength for a database. It truly is. If you can make changes and then deploy it to production and then get the traffic of billions of users, you find problems very, very, very quickly. And that makes things very strong. Look at Facebook's adoption of MySQL, the work they've done from MySQL. It just helps. It's just so, so much better. And that is a real strong suit for Vitesse.

[00:12:59] JM: So PlanetScale recently made it to general availability of the hosted database. Can you define general availability and what it meant to get to that place?

[00:13:13] SL: So six months, we were in beta. And it was really about continuing to build out some of the central systems we needed so that we could feel we could go GA. Kind of having users stress test the system and make sure that it works and it does the fundamental things it's meant to do. And there's so much trust involved in a database. You've got to take it really, really carefully. And we take it really, really seriously that we're the custodians of people's databases. And then, by proxy, it's their business. And if there's any database issues, it's real hard. And we take that super seriously. So we wanted to be – The database world is swamped with snake oil,

and we didn't need to add more hubris and recklessness to the mix. We wanted to take things very slowly.

But to our surprise, people started moving really like decently-sized workloads over to us in beta. And things started to work really well. Like that very large website that we talked about moving moved during our beta. And it went fantastically well for us. And we thought, “Well, if their availability is improved moving away from Amazon RDS by moving to PlanetScale, like we're close to ready.” So we started to prepare – The engineering team did a phenomenal job. They all came together and made a massive list of all the things that need to be better, need to be improved. All of this like rough edges and snags that we had. And they just went through it, got it all tied up. And we shipped. And I was really surprised that it was only six months. And considering the platform came together and the first commit on building the platform, like everything around Vitesse that we put together to build the product didn't exist before December 2019.

So we've moved an incredible pace. We went from nothing, like literally – Like I was going back through the demo videos in the early days. And about this time last year, we had like a blank login page that you could log in. Like there’s just nothing. So in less than a year we went from not just built, put it out into the market with features nobody's ever seen, rocket ship user growth, and GA in less than a year. It was just an awesome year next year. And then a lot of it was getting to fundamentals, like a lot of undifferentiated things.

Single sign-on compatibility, and audit logs, and user, kind of password reset flows, and all of those little things, undifferentiated. Everyone has to build them. And I'm super excited that we have a year ahead of us now that we've done all the fundamentals. We've got the basics out there. And now it's kind of pure, cool stuff. Like this year, our roadmap has got some incredible things on it that we’re really excited to show people. And it’s stuff that people haven't really seen databases doing.

[00:15:50] JM: Like what?

[00:15:52] SL: So I'm not going to go and sneak the roadmap out there. But one thing I will say is that Vitesse played many, many roles at YouTube, and did a lot of things that are not just

being the MySQL compatible database. There is an immense amount of power and robust power under the hood. And our journey now is to kind of start surfacing that power to people.

[00:16:12] JM: So you worked at GitHub for a pretty long time. How did the database challenges at GitHub compared to what you've seen with the customers of PlanetScale?

[00:16:25] SL: Depressingly, it's all the same. There's a large set of problems that hold the whole industry back. And these companies that spend a massive amount of time solving database issues, when they should be doing things that differentiate their business and help their business grow. And I think there's these last years of – We speak to so many, so many customers that are victims of their own success. And it's the database that's letting them down.

Very often, it's like the CEO comes to these conversations. Because when you're having database problems and outages or really slow development cycles, like, it's a top level priority. And I speak to so many people at these companies that are going through really difficult issues, and there's no way out unless you kind of re platform and move to something else. It's really tough. And we were in that same spot at GitHub. We grew wildly successfully. And the things that always broke first was the database. And we had a fantastic database team that spent a lot of time and built a lot of tooling to make it manageable. But it was never fun. You were always catching up.

[00:17:29] JM: Do you see PlanetScale used for OLAP workloads, or just for OLTP?

[00:17:37] SL: Hybrid of both never works. I know people are trying it. But it doesn't. So we really optimized to be really good at OLTP. And we're going to give people easy ways to get the data into their OLAP databases. And we use OLAP databases for our own analytics and data gathering. Right now, we're purely focused on being fantastic OLTP.

[00:17:58] JM: Vitesse, it's a piece of technology that's not necessarily married to MySQL, as I understand it. Do you imagine a world where PlanetScale offers, I don't know, Elasticsearch backed by Vitesse? Or, I don't know, Redis, backed by Vitesse? Do any of these other database use cases make sense?

[00:18:20] SL: We talk about it. And it's interesting. I think Elasticsearch would be too much of a stretch. The one that we get asked for all the time, I mean, like, nearly daily now, people ask us for Postgres support. So there is a lot of – The architecture, yes. It's not necessarily fully bound to MySQL. But there is a lot there. Like MySQL, provides really good like replication primitives and ways of doing things. And the fact they're reliable is a thing that makes Vitesse very reliable. But Postgres is the one we get asked for constantly, like all of the time. And we consider it. We talk about it. But it usually comes down to it shouldn't matter. Like, the backend, the exact backend storage engine, whether it's MySQL or Postgres, shouldn't be up for concern in the long run. Really, it's going to be about all of the other incredible things that we're going to do, and whether your database platform can do that, versus whether it's Postgres or MySQL in the back.

[00:19:19] JM: So, PlanetScale, obviously has this core innovation of database scalability from the tests. And beyond that, you've built on these database usability features. What do users expect out of that database admin and usability layer? And what are the opportunities for innovation there?

[00:19:43] SL: I think they expect being able to do things quickly and intuitively without reading miles and miles of documentation that kind of explains every caveat and weird side case or whatever the database has. I think people expect to trust the database to do what it's supposed to do and that it does that very, very reliably. So that's why we really think about what our job to be done is. And we kind of refine our understanding of that as time goes along.

But I think people just – Actually, like most people just don't want to think about it. It's very clear what the database is there for. And I think people feel – They feel betrayed almost when it doesn't do that. And we really want to make sure that we do exactly what we're meant to do. So I think that usability sounds simple. It sounds like, yeah, of course, everyone should do it that way. It's not as simple as that. Like, being able to do complex operations from a CLI that is intuitive and easy to read is not an easy UX problem.

And when I compare ourselves to other tools and see like – So, for example, like we're no less secure than any other platform. Like we provide you an encrypted connection to MySQL. But we've made sure that the certificate exchange is handled by the CLI and that is made super simple elsewhere. Whereas other people are like, “Download this. Stick it in this folder. Do this

thing.” And it's just like unnecessary waste of time, complexity that can cause issues for people that are trying to learn, or begin, or get things. So we just spend more time refining it making it super, super simple and don't assume much knowledge from the user.

[00:21:23] JM: What's the feedback loop between talking to customers about what's working and what's not, and getting those features built into the admin situation? Or I just love to know more about the feedback loop for getting features into PlanetScale.

[00:21:46] SL: Yeah. So we talk to our customers a lot. I get very involved in the sales process, so does our head of engineering. And we listen to things that may, say, block a sale, or that really hamper a customer having a good import experience or moving out to PlanetScale. And we look at those things.

I think the core reason we have this product that's as good as it is, is because our engineering team really care about the users. So I spoke to someone who was a founder of a company that's already doing extremely well. And they said, “Oh, just –” Now we’re talking about something completely unrelated with them. And they said, “Oh, just you know, like, we’re PlanetScale users.” I was like, “Oh, that’s great.” And he’s like, “And I tweeted the PlanetScale account that I had a problem. And three of your engineers DM’d me all at once. And that was just a really unusual experience. I've never had an experience like that.”

And I was really proud of that moment. Because one thing I've noticed with our engineering team is when they get more sources of information or feedback from users, they dissect it really quickly, and really think about it. So we don't put walls up or gates between our engineers having genuine conversations with other engineers like themselves to understand their problems. We don't have product managers for that reason. And that's really, really important. Everyone has the freedom to go out and talk to us as an advocate for users internally.

[00:23:07] JM: Are there any particular problems that you've seen customers encounter in MySQL sequel deployments that you simply don't have a good solution to? Like things that you're not able to solve in the product? Like, just are there problems inherent in MySQL that you simply can't solve?

[00:23:26] SL: So far, no. If they're on MySQL, they've likely had problems with things like scalability with MySQL or performance. You do come across certain people that just shouldn't be using a relational database for the things they're doing. But there's heavy bias with the fact that we sell MySQL OLTP solution. So we spend up – I end up talking to customers that use that and have that as a problem.

Sometimes there is real issues with people's applications, and they're doing things in wonky ways. And you kind of have to break it to them that it's like, “Yeah, really, like, this isn't going to work in any sense.” And you kind of coach them through changing their application. But we get used to that. That's kind of life as a database company. And the more we hear those types of things, the more we think, “Well, there's product features here. There's intelligence. And there's ways of surfacing these things to the user without our help.” That it's just all opportunities for us to improve the product.

[00:24:21] JM: When I read through the PlanetScale blog, there's a lot of posts about database internals and nuances of certain MySQL operations you can make that take place under specific – Or you want to use under specific circumstances. How much do you personally have to understand about MySQL internals as the CEO? Do you feel like you can just kind of like learn as you go along? Or have you had so much experience with MySQL that you really have a deep familiarity with a lot of the subtler operations?

[00:25:01] SL: My background is a MySQL DBA. So I have a lot of experience with MySQL. I haven't done it for a long time. So I haven't kept up to date with every single feature or nuance change. But I know how these things work. And I could describe pretty simply the architecture of MySQL and why it's good at what it does. But it's been a while since I've spent a whole amount of time with a large MySQL deployment. But it's enough to know what we're doing. It's at the appropriate level. There's definitely things and tradeoffs in MySQL that can make things complicated.

But, yeah, I have a good understanding of that. Still build small applications using MySQL, and I've kept pretty much – I’m excited about it. It’s databases. I love it. So it's still just a high-level of interest for me, and I keep up to date with what the engineering team are doing, and then try not to medal. But just keep an understanding and have a high-level of appreciation for it.

[00:26:01] JM: Is there a specific technical feature of MySQL that you've learned recently that you could share? Is there anything you've learned from the team?

[00:26:10] SL: Vitesse. I've been really learning about the power of V replication in Vitesse. So the way Vitesse works is it's very good at replicating data between disparate MySQL machines or just disparate shards of MySQL. And that's a kind of hard problem to solve. And there has many, many possibilities for what you could build with a system that is that robust. So I've been spending a bunch of time learning about that. And that feels like a treasure trove of just great tools and software that still has undiscovered by the wider community. And so I spend my time learning that, and still getting pleasantly surprised each time.

[00:26:53] JM: I'd like to switch the conversation talking a bit about engineering management. Is there a particular book on engineering management that changed your perspective on the practice?

[00:27:05] SL: No. Honestly, I think my engineering management is one of the things done absolutely –One of the most suffering parts of the tech industry is the craft event of engineering management. I think it's done so badly at most companies that it really lets people down, especially engineers. And I just try not to get involved in the sense that there's probably some great books out there. There's probably some really good techniques and practices. But a lot of is generalized. And once you get the basics down, I think people spend too much time being professional managers, versus people that are there to support a very creative process that should result in great engineering being done.

I think the right engineering managers view themselves as a kind of necessary evil on the path to getting great engineering done. And those are the best ones. And unfortunately, there's a whole – And the majority of engineering managers interesting – So the industry-wide kind of opinion of themselves is that they're there as some sort of special, massively necessary role. And I just don't fully agree with that.

[00:28:16] JM: Can you say more about that? I'm not exactly sure – What is the anti-pattern of engineering managers that you're really honing in on there?

[00:28:24] SL: Engineering managers tend to find their work as being – And like as a result in itself, like how many engineering management time working on reorg and ways to neatly bucket their teams into nice little slots, versus being able to kind of support a highly creative, chaotic, dynamic environment, and letting great people do their work?

I've worked with, unfortunately, too many of these people that don't spend their time learning the craft their teams are building and learning the craft of engineering and supporting good engineering being done. They just push paper around and do this kind of middle management, which just wastes massive amounts of resources and really does nothing except grind businesses down. We have a very different management philosophy. We want the company to be the best place to work. And we don't think managers that come to collect resources – When they call people resources. Or control information to gain power. And I don't know. As soon as you start hiring professional managers that are there to just manage people and not get work done, politics comes into the organization, and it just becomes a mess.

[00:29:38] JM: I wrote a book about Facebook engineering. And when I think about the Facebook engineering, the product is so expansive and has so much surface area. It's very easy for me to imagine this kind of decentralized, highly creative work environment that you're referring to. But when I think about a database product, to me, a database product is so mission-critical that you almost – It seems like you would want a rigid engineering organization. And you would want engineering manager – And this is maybe a caricature. But you would want engineering managers that are kind of like taskmasters, thinking very rigidly about what needs to be done. Why is it that a database product is something that can be creatively hacked on?

[00:30:27] SL: So, no matter how difficult, and low-level, and mathematical the solution to a problem is, it takes creativity to solve all problems. It takes optimism. It takes being inspired. Have you tried solving a hard problem without feeling that even solving it is beneficial to you or something that inspires you? You'll never get it done. You can achieve both things. Like, don't get me wrong. What you've been talking about is – And I worked at Facebook, and I encountered some of the best managers I've ever seen in my career, especially upper leadership in the engineering org, were just phenomenal. And they achieved the same level of high – The way Facebook internally talks about the infrastructure and the crushing weight of that scale of billions of monthly active users. And they're very hard infrastructure problems to solve. They take it extremely seriously.

But you can – Yeah, you can have managers that like hound people, and drive them crazy, and bug them, and create endless reports and metrics that don't really mean anything to get some level of rigor and robustness. Or you can have managers that inspire the want to do that in their people by being just as good as their people, maybe even better. And they're able to perceive a better version of the people they work with and that they support. And they inspire and drive people to be their best selves. And that takes competence. And that takes mastery of the craft.

So at PlanetScale, to be a manager, you have to spend your first six weeks programming. You do not get away from that. And we stole that from Facebook. They do the same. But when you saw our VP of engineering, he came in. He spent six weeks engineering with the team building. And it gave him a such an appreciation of the work his team has to do. And that gave them respect for his engineering ability. Where you're seeing a lot of organization, the VP of engineering, "Oh, I haven't written code for 10 years. But I'm really good at resource allocation." Well, I mean, yeah, they may be able to micromanage people to death to get a similar result. But you can be just as good as inspiring people.

So, Jay Parikh ran the infrastructure team at Facebook. He was a phenomenal leader. Phenomenal. And part of that leadership came from competence and people knowing that he could probably do the job of most of the team below him just as good or better. And so it's just different. You lead from the front or you push from the back. It's just a different style, in my opinion.

Can I read you like a little bit from my manager expectations? So I think I can make it more clear to you.

[00:33:07] JM: Sure. Yeah.

[00:33:08] SL: So you can expect your manager to perceive a better version of you and support you in getting there. Gather and deliver relevant signal an opportunity for impact tailored to you. Provide clarity of purpose of direction and responsibilities. Managers should be capable of making strong technical contributions, even though they may not spend much time doing so. And they can drive a culture of excellence and attract increasingly talented contributors to the team. And your manager will not micromanage or coddle you. Instead, they'll insist on a high-

trust, ownership-oriented culture. They won't command the title. Instead, they'll lead with influence. They'll serve as a proxy or firewall. They will not serve as a proxy or firewall. Instead, they'll support you in building healthy, direct lines of communication with your colleagues. They won't pay politics or hoard people. And they'll incentivize people to put like the team and customers first. And the last is my favorite, they will not fill their time with milk toast busy work. Instead, they'll enthusiastically engage with the work of getting things done. So that's the difference. We have a longer list of things they won't be doing, versus what they should be. But to me, it's more about leading people and getting a great experience for them to do their best rather than pushing people around.

[00:34:21] JM: And do you have any systems of accountability or ways for tracking progress methodically that may not inhibit that kind of looser, more creative situation that you'd like to create?

[00:34:38] SL: We're still small, right? We're still only 85 people. So we don't have tons of systems and processes for doing these things. And we try and avoid them. It’s largely subjective. Like, everyone on PlanetScale knows their craft really well. And we know when things are taking longer and things are slower, or more complex, or more noisy than they should be. And we just drive out really quickly. We have a very open, frank discussion about the issues. Just get to resolution.

We only have one sort of value at the company, which is everyday matters. And we selected it because we found ourselves living by it. I think most company values are nonsense. I think if your company values excellence, you don't have excellence. You have the opposite of excellence. If it has to be demanded or asked of your people, then you've hired incorrectly. Everyone at PlanetScale comes to work because they want excellence. They want to close their laptop at the end of the day and know that they're contributing to and building something very special with other very, very smart people. No value written on the wall of excellence, or empathy, or any of these like just generic terms that people use as their company values does anything for that. It's about culture. It's about who you hire. And most importantly, it's about firing people that either disagree or don't live up to that way of working.

But so anyway, I digress. We have one value, which is everyday matters. And what that really means to us is do things now. Optimize for making today the best day. And that losing time and

wasting time on putting things off for a week away is just a waste of time. We optimize for an extreme bias to action of making things happen. And people start to get very frustrated very quickly if things don't move. And that's good. And I do not ever want to just like extinguish that fire for fast progress.

[00:36:33] JM: Do you have a rigorous test suite for changes to the database? Or do you guys – Are there a subset of the surface area of the database that you test more rigorously?

[00:36:45] SL: Yes. So we have – Yeah, it's very, very robust test suite. We test against all major frameworks to make sure that they don't see regressions or compatibility issues. We performance test. You can move fast and take things very, very seriously. So the Facebook motto of move fast – Used to be move fast and break things. But they changed it to move fast with stable infrastructure. And I think actual stable infrastructure, reliability in tests are precisely how you move fast.

I deployed facebook.com in my second week at the company. Like I pushed code into facebook.com. And I couldn't believe how incredibly like simple the process was, because of a ton of really good infrastructure and automation that makes it like not at all scary. We want it to feel the same when you're making database changes. So we are extremely rigorous with our changes. And it also helps that some of the largest websites on the Internet use our software. And when we do a release, they take it. They test it themselves, and they put it into production. And we have contributions from pretty much all the major websites in Vitesse.

So it's really standing on the shoulder of giants and getting tons and tons of like – Every Slack message in the world is in Vitesse. When you send a Slack message, it gets acknowledged by Vitesse before it actually gets sent around to the client. So it's very robust in terms of the amount of traffic and testing it gets. And that is really, really handy for us. It's not just us deploying it. It's other large customers.

[00:38:12] JM: To get into a particular engineering question, the Kubernetes clusters that run on the managed PlanetScale instances, can you tell me anything about those Kubernetes clusters? Like, are there any particular, like – I don’t know. Service mesh, or operator patterns, or anything you're doing interesting in those Kubernetes clusters?

[00:38:36] SL: I'm, by no means, an expert when it comes to Kubernetes. I'm told it's fairly vanilla in terms of the Kubernetes set up. And perhaps we could have someone from our infrastructure team come in and talk a little bit more about it. But we try and keep it simple so that we can deploy into the managed environment, right? And so that it's kept – Yeah, I don't have too much detail about the depth when it comes to – I leave that to our fantastic infrastructure team.

[00:38:59] JM: What about product design? It seems like getting developer experience optimized has a lot to do with design. Do you involve yourself in design review meetings? And do you guys have dedicated designers? Like what's the process of getting design optimized for a database?

[00:39:18] SL: So we bring designers in at the very beginning of the conversations. If it's going to be touched by users, then we have designers that have built fantastic developer tools. In fact, our two product designers report to me, and every week or month I check with them, “Are you sure you still want to report to me? I'm not very good and not as engaged as one would be necessarily being the CEO.” And we all agree we wouldn't change it because it's so important to have designers at the seat of the table even with backend engineers. And it's the only way you create great user experiences. You kind of explain what the job that the user is going to need to do. What the job to be done is? What the user story is? And you have the designers work on it. And it's not just thrown over the wall at the very end to be like make this thing pretty, or like we don't have like backend engineers doing mocks of UIs and then designers just get like two weeks before launch to make it pretty.

Right at the beginning, like we just had a number of – We call them Zoom sites, because we didn't do them in person. But like offsites, basically on Zoom, where we were kicking off the major features that we're delivering this year. There's a designer in the room for every single one of them. And they're already working on mocks. And the mocks and the designs are being done in parallel to the engineering work to make sure we were fine both. And they meet in the middle in an acceptable place. If you have backend engineers designing stuff, they invariably come out with things that are a little bit more complicated, because they understand the tradeoffs differently. And that's not wrong. It's fine.

The job for us is to kind of keep a healthy tension between the two of this thing should have no steps or very few steps to execute this journey. And it should be heavily designed. And it should be achieved by magic in the backend. It's the same with our brand designers as well. We have in-house brand designs. I get asked every single week, "Which agency designed your website?" And I always say, "They didn't. We have people in-house." We have people that put design on every single blog post we do, every piece of social. Like it's extremely important to us how our product and how our brand is represented in the world.

[00:41:19] JM: Are there any other infrastructure companies, modern infrastructure companies, that you take inspiration from? I was going to say, your website looks a lot like Vercel, in a complimentary way.

[00:41:30] SL: Yeah, of course. No. I think they're a wonderful company. I think there are people – They remind me very much of PlanetScale. And people say that PlanetScale is the Vercel for databases. And that's right. And again, very complimentary. And we share a massive community together. I think it's becoming Vercel, PlanetScale and Prisma combined is becoming a pretty formidable stack. And I see people using it constantly. And there's already some pretty successfully big websites running on that entire stack.

Yeah, I think whenever you meet someone from Vercel, they universally agree that they should be building something incredible and usable for their audience. And it's the same with us. And when we all do get together and hang out, it's just a melding of minds. It's awesome. And, yeah, I just have an immense amount of respect for what they've done and continue to do.

[00:42:16] JM: That stack of Prisma, Vercel, PlanetScale, what are the advantages that that gives you like in terms of actual day-to-day usability? Like when you think about a comparison to much more traditional stack like, like the LAMP stack? I mean, this is like three cloud native technologies compared to three open source technologies from back in the day. How would you draw that comparison?

[00:42:44] SL: So first of all, I've still got immense love for the LAMP stack. LAMP is going strong and keeping a huge amount of the Internet together. But I think to answer your question properly, the combination of the three gives you incredible leverage. And that is that it's fundamentally at its base what infrastructure should be about, leverage. And you should be able

to leverage good infrastructure, so that it adds time to your day. Not subtracts, right? It gives you a repeatability and a scalability. That means you don't really have the same problems over and over again. Like if you get your stack and architecture right early on solutions like these, you have an incredible amount of scalability ahead of you. And that means you're not rewriting how your database code works. That means you're not changing database from Postgres to MySQL. It means you're moving fast and continuing to grow and growing your business. And that cannot be underestimated.

Like the years of engineering time – I spoke to a customer. They're huge. Like, they're migrating to us. But they're massive. Really big, really huge MySQL deployment. And I couldn't believe their answer when I asked them. I said, “How many years do you think you've lost to database scalability problems within your company?” They've been around for like 10, 15 years now. And with a fully straight face, he said, “100 years.” And I was like, “I couldn’t believe it.” I just couldn’t believe it. I was like, “Maybe 10.” He’s like, “No. We’ve got a team of about 10 people that work on databases full-time. And it's just a constant issue.” And that's crazy to me.

And I think people have the same with like frontend scalability as well. Maybe not as bad as databases. I think, peaks at databases. But they have these issues. And so, when you pick a stack that's built on these types of platforms with very little backend knowledge that you probably don't need to develop the muscles for a long time in your company, you can get extremely far.

[00:44:43] JM: Well, just to wrap up, I'd love to get your perspective on what has been hard about becoming a CEO. Like, just what are the hardest elements of managing a database company?

[00:44:56] SL: I don't know if it is specific to database companies. And I think I probably have an easier job than many based on how great the team of people is here. And I think we certainly have avoided a lot of the problems companies go through by just having a very high bar for the people we have at the company. So I feel very lucky.

Almost to the hour of like becoming – When you become the CEO, there’s strange kind of just feeling you get up like, “Well, the buck stops here.” Like everything is potentially your problem. Like when you work at a company, like you work in the engineering team, you're not worried if

the bills get paid. You assume someone's paying for the office. Someone's job, somewhere to do something. If you're the CEO, if there's that person, they're not doing something, and it's causing problems, well, that's your problem. It eventually rolls up to you until you can find a more sustainable way to fix it.

So you get this kind of strange feeling that is the unknown unknowns are your problem, and you don't know it yet. And so you have to continue to wait. It's not something that I feel can undermine your thinking or make you overly paranoid. But it's certainly a sense of responsibility that goes a layer deeper than any other role I've ever had. And you talk to other CEOs about it. And we all agree, it's a pretty lonely job.

And sometimes doing the right thing can make your – I mean, you're, yeah, kind of everyone's enemy for a little while, or you have to push for certain things that don't make sense to everyone immediately. And then it's your job to explain it, because you owe everyone that. And so it's a lot of additional responsibility that I never quite expected. But at the same time, it's an amazing job. And I feel it's a role of servitude towards building a great team with great people and a great company. And it's something I'm more than happy to take on.

[00:46:44] JM: Awesome. Any tips for getting those problems, those unknowns, to bubble up?

[00:46:50] SL: Listen, pay attention, and always have an open door to people to come and talk to you. There's always someone that's bothered by something. And if they have to bottle it up and not get it out and not communicate with you, that's when problems fester. If people know they can come to you, and you'll deal with things super quickly, and you'll be grateful for them raising the issues, then things come forward. Certain stuff you'll never predict. And you have to have a sense of humor about that. I mean, some of the stuff that happened in our journey at GitHub. It was just kind of funny. In the end, you think, “Well, I could never have predicted that.” And kind of just you roll with it. You have to have a sense of humor about it. But just having a team of people you're close with that you listen to, really important.

[00:47:31] JM: Well, Sam, thank you so much for coming back on the show.

[00:47:32] SL: Thank you so much for having me. It's always a pleasure.

[END]