EPISODE 970

[INTRODUCTION]

[00:00:00] JM: GitLab is a company that builds an open source platform for managing git repositories. GetLab was started in 2012 and it has grown to have a large enterprise business with additional products, such as continuous integration and security tooling.

GetLab is also known for building a large, entirely remote workforce. GitLab does not have any offices, and the employees mostly communicate through Slack, email and GitLab itself.

Marin Jankovski was the first full-time engineer to join GitLab shortly after the company was started. Marin joins the show to talk about the very early days of GitLab and the evolution of the remote culture and how product development works at GitLab today.

GitLab is one of the fastest growing software companies, and this show was a great exploration of why that is. It explain some of the philosophies behind the company and how operations work. It's a great episode and there's also some discussion of the public spotlight that GitLab is in and the impact of being in the public spotlight.

We had some discussion of the controversy and the debates around the somewhat political issues that GitLab has found itself in on the Internet. These were not big issues, but they received a lot of publicity, because GitLab is growing so fast and it has had to reckon with some of these difficult decisions that any fast-growing software company has to make.

Speaking of remote work, we are hiring for two roles, a content writer and an operations lead. If you like to write about software engineering and study software engineering, this could be a great role for you, the content writer that is. Separately, if you're looking to understand how a business works, or specifically a software engineering podcast business, then the operations lead role might be for you. These are both part-time positions working closely with myself and with Erica. If you're interested, send me an email at jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:02:19] JM: If you are a SaaS or software vendor looking to modernize your application distribution to gain more enterprise adoption, checkout replicated.com. Replicated provides tools to deliver your Kubernetes-based application to enterprise customers as a modern on-prem private instance. That means your customers will be able to install and update your application just about anywhere.

Bare metal servers in a cloud VPC, GovCloud and their own Kubernetes cluster, vSphere. This is a secure way the your customers can use your application without ever having to send data outside of their control. Instead of your customer sending their data to you, you send your application to your customer.

Now, this might sound difficult and maybe you’re not used to it because you're a SaaS vendor. You’re a software vendor, but Replicated promises that recent advancements from tools like Kubernetes make it far easier than before, and the Replicated tools can help vendors operationalize and scale this process.

The Replicated tools are already trusted by noteworthy customers like HashiCorp, CircleCI, Sneak and many others. As a result, over 45 of the Fortune 100 already have an application deployed via Replicated in their infrastructure. That’s a strong sign of adaption.

Go to replicated.com for a 30-day trial of the full Replicated platform. You can also listen to an interview with Grant Miller, the CEO of Replicated, that we did a while ago.

Thank you to Replicated for being a sponsor of Software Engineering Daily, and you can check it out for yourself at replicated.com and get a free 30-day trial.

[INTERVIEW]

[00:04:27] JM: Marin Jankovski, welcome to Software Engineering Daily.

[00:04:29] MJ: Thank you. Thank you for having me.

[00:04:30] JM: You joined GitLab in 2012. Describe the company when you joined. What was GitLab like back in 2012?

[00:04:38] MJ: It is very simple to explain actually. The company got formed in August 2012, and I was working with Sid, the current CEO in his previous startup. The company was – Well, almost nonexistent, because one day Sid called me and said, “Hey, I have a great idea. I think it's a great idea. Want to join?” I was thinking why not, because one startup or the other startup, does it really matter? Right?

The idea was actually really interesting. The first thing I told him though was, “You do know you're going against GitHub, right? It's two of us here.” He’s like, “Right. Yes. But we’re going to build a better tool.” So this was enough, actually, and it has been a wild ride since then.

[00:05:24] JM: That sounds like Sis. I mean, people who don't know can go back and listen to – We did an interview with Sid, and I was definitely struck by his personality. He is one-of-a-kind. I will say that. He's very ambitious, but very soft-spoken. What is it about Sid’s personality that makes him a good CEO?

[00:05:48] MJ: He knows how to say sorry a lot actually. As I told you as an intro to this conversation, he's very opinionated and he has a vision of how things need to be. What makes him an amazing CEO though is when you confront him, he is willing to listen. He’s not strong-headed when it doesn't matter, which a lot of people actually make that mistake, right? They allow ego to drive them. He doesn't seem to have much of that actually.

So the first three months of us working together prior to GitLab were extremely rough for both of us. We come from different cultures. We come from different – Completely different backgrounds, upbringing and so on. The first three months were both of us like just clashing constantly and we did not see the end to that.

As some points my girlfriend at the time said, “You know, this is making you miserable, right? You are constantly upset. You are constantly saying that you’re not sure whether you want to work on this anymore, but you're not taking an action. What's happening there?”

I didn't know at that time, but Sid’s now wife said something similar to him. So at point, I kind of threw my hands up in the air and said, “You know, I tried my way and it hasn't worked. I'm just going to try to be open about it and say, “Yeah, I don't think this is actually working. I’m pretty miserable with our collaboration.”

I think that kind of opened the door for us, because we had a really long, frank conversation, very direct. Kind of extracted. I think both of us kind of threw our hands up in the air and said, “You know, let's see what we can make out of this,” and that set a completely different path for us. Completely different path, because we kind of started trusting each other and believing that the other person is going to do what's best. Then when the other person doesn't, you just approached him and confront him and say, “Hey, directness. I'm going to be perfectly frank.” You know, putting a red bow on it, and since then, knock on wood, it has been amazing journey.

[00:08:00] JM: Did you both changed at that time?

[00:08:02] MJ: I for sure has have. I think Sis also changed as well. Then some of our core values were born out of those conversations, like trial and error, transparency iteration, all of that has been born out of our initial collaboration there. So I didn't know how to – I didn't even know do meaning of the word iteration. I was an engineer at that time and the only thing I wanted to build was a perfect solution. He was an engineer as well, but he was more business-oriented. The only thing he wanted to do is beat the market. So together we kind of learned how to do this step-by-step and hit the market faster, and iteration is now ingrained in GitLab.

[00:08:47] JM: Beating the market. So conversation with Sid, I do sense a fierce competitive streak to him. He's a soft-spoken, quiet, fierce competitor. You could tell. Why is that useful? Isn't it better to be positive-sum? I mean, Bezos is always saying, “We don't think about competitors.” Why should you be thinking at all in competitive terms? Why do you want to beat the market? Why don’t you want to just redefine the market?

[00:09:19] MJ: At that time, I'm going to remind you 2012. We were going against GitHub, a tiny startup of two people going against a mammoth company. So at that time it was important for us to ship fast, ship very fast.

At this stage of the company though, I think we are big enough that we are starting to set trends. I don't know whether we can be called trendsetters, but we are doing certain things that are definitely not in line what the market does. Like 2015 when we merged our CI into our main product. No one thought that was a good idea, including Sid. He didn’t also believe this was a good move. But at an engineer at that time that was in the company said, “Well, the fact that the market says this doesn't mean it's the best possible solution for us. So how about we try and do this this way?”

So things change as you grow obviously, right? As you start being like more positioned. But one of the other bets that we made was betting in Kubernetes when it was not at all clear that Kubernetes is going to be the winner in this container platform war, right?

So I vividly remember Sid coming back super excited when he read the Kubernetes announcement saying, “We need to watch this. Something is going to happen here,” and he had been watching that ever since. I think back in 2016, maybe beginning of 2017, he said like, “This is a done deal even though the market was not saying that.” So he has this streak of vision in him. He’s able to carefully study things.

[00:10:56] JM: Okay. So you guys were the first two people in GitLab. I thought there was – Were you the guy who started GitLab? Who's the one who started the open source project?

[00:11:05] MJ: The open-source project was started by Dimitri. He’s from Ukraine. So he started in 2011 when he created the project, but he joined the company at the beginning of 2013. So when I say we, I mean Sid and I. We worked together on contributing back to GitLab. Trying to stand up gitlab.com as a SaaS offering at that time, and Sid the reached out to Dimitri asking him or rather telling him, “Hey, I'm going to use your open source project. I'm going to try to make money out of it.” Dimitri was receptive to that and said, “Good luck.”

Over time, or six months actually, we kind of build a relationship with Dimitri through contributing to GitLab as the open source project to finally have Dimitri Tweet the famous tweet, “I would love to work on GitLab full-time,” and that points, Sid reached out to him and they worked out a deal that made Dimitri join in January as a cofounder.

[00:12:03] JM: 2012 was pre-Slack, the early days of GitLab. Now that said, I guess you were a two-person company. So remote team of two people, not two – Were you living close to him?

[00:12:16] MJ: No, not at all. Sid was living in the Netherlands. I was living in Serbia. Serbia is my country of origin, and we used to Skype.

[00:12:24] JM: Oh, yeah. That's a great one. Great co-working tool.

[00:12:27] MJ: Can you remember how that looked? Yeah, we screen shared –

[00:12:30] JM: It actually does the trick, right? You can have like three-person chats, and four-person chats and multiple person group calls. I guess that does most of what you need out of a group.

[00:12:43] MJ: Two people, three people, it worked. Dimitri joined, then we switch to Hangouts. Remember Hangouts?

[00:12:49] JM: Of course. I still use it occasionally. One of my friend started messaging me in Hangouts again, like, "Come on, man. Not Hangouts. I got enough channel. Nobody uses this anymore." Well, some people do.

I guess when the company really was scaling by then, did you have Slack, or what was the remote team management story?

[00:13:08] MJ: We switched a lot of options over a period of time and we were on Campfire. We were HipChat. Remember HipChat?

[00:13:16] JM: Of course. Yes.

[00:13:18] MJ: We were on HipChats as well. Slack came I think very late into this story, end or mid-2014 actually. If not end, of 2014. At that point we were already I think five or six people, and Slack was the new kid on the block and we were like, "Okay. Let's give it a go at that time." I think there were – I don't know if they offer still free plan.

Yeah, the tooling we used at that time was Google Docs, Skype and Hangouts, and we went through multiple chat options. So slack was only 2014, 15.

[00:13:54] JM: do you feel like there are gaps in the tooling today for remote team management?

[00:13:59] MJ: That's an interesting question, because I've been thinking about that very recently. I don't necessarily think there is a gap in tooling. It is the gap in how this technology is not mimicking the standard human behavior. What I actually mean is Zoom is an amazing tool. We use it exclusively now for our communication.

What is missing in technology at the moment is that natural conversation between multiple people that can happen while you're standing in the same group. When you're talking in Zoom, you always have one conversation. When you're in a group in real-life, you can have multiple conversations at the same time. So that is actually missing. Everything else, I think we are well-equipped to lead a company, and GitLab is proving it, that it's possible to do. Just that human aspect of having multiple conversations within one larger conversation is what I think is currently missing.

[00:14:57] JM: Have you tried that thing Tandem, the co-located – Or it's like a virtual – Basically, it's like a video chat room that's there all the time.

[00:15:07] MJ: I have not. No.

[00:15:07] JM: I haven't tried it. I just heard about it. Explain how to GitLab makes money today.

[00:15:12] MJ: So majority of our income comes from our enterprise offering. We have an enterprise edition with multiple tiers. That enterprise edition – Well, depending on the size of your company or the needs of your company, has more or less attractiveness. For example, our starter offering has some additional features compared to community edition, but the main draw for it is support that we offer for our customers. Then the ultimate one, which is the most expensive plan we have, is definitely geared towards more developed enterprises that need end-to-end solution for security, for CICD and various other things.

We also do have gitlab.com, our SaaS offering that has comparable tiers as well. Gitlab.com is free for majority of users, but you can still purchase a plan that will unlock certain features for you. So Gold plan, for example, will offer you epics and issue boars and so on and so on. But that is a smaller chunk of our revenue. Definitely, enterprise, self-managed is where we get most of our money from.

[00:16:22] JM: Has it turned out that you are competing with GitHub, or did it turn out that you expand the market or found new areas of the market that GitHub was not even exploring?

[00:16:34] MJ: At the moment, we started thinking about CI and started building our CI tool, and then integrating into our product. This is where the story changed for us. Up until that moment, it was Bitbcket, Atlassian, GitHub as source code management tools. When we integrated CI tightly inside of GitLab, the story changed for us. We started marketing ourselves as complete solution for not only source control management, but also for deploying for testing and so on. I think that is the point where we started – Let's call it sprinting instead of walking when it comes to that race with GitHub and Atlassian.

[00:17:20] JM: And that was because, as it turned out, there were many people who wanted to move into CICD. They had trouble figuring out how to integrate everything together.

[00:17:31] MJ: Correct, or spend much time integrating things together to make it work.

[00:17:34] JM: Right, like taking CircleCI off-the-shelf and figuring out how thins puzzle piece fits into GitHub. That was more work than just going with GitLab, which comes out of the box fully integrated with CICD. So the controversial decision to integrate CICD ended up being the windfall decision.

[00:17:54] MJ: Correct. Correct. Then at certain points when we started talking with Gymnasium, so security scanning and so on, same – Let's call it a problem, appeared where all these very advanced tool are also very advanced to manage. What users actually need is the simplicity and they need one single interface. They need to focus on their own thing and not on the tool itself. Integrating these solutions and like shifting them left towards the actual developer has changed how we tell our story.

[00:18:30] JM: Okay. So CICD being integrated with the code hosting repository, that makes complete sense to me. What does a security product have to do with that?

[00:18:41] MJ: So how does a regular development, delivery workload look like? Developer commit some code, does some testing that gets merged. Gets deployed, and then it goes to a security departments that verifi es the changes. Make sure that everything is okay, and that means like we are now ready to release.

[00:19:00] JM: Then they approve the rule out.

[00:19:01] MJ: They approve the rule out. Correct. But the problem is you already packaged your software. You already had spent a lot of cycles making sure that your codes is working. Now, if the security department comes back and says, "You need to redo this." You need to repeat the same process again."

If you move your security scanning all the way back to developer and give him immediate reaction to whatever they have done with their code, they can fix it right away. They don't have to repeat the process.

So you give not only the developer, but you also give the security department and of the people involved in that process more confidence that in vast majority of the cases you have covered what is necessary for your software to be delivered. Obviously, in some cases, you still need security to approve certain changes. But now they're focusing on high-level risk versus you didn't update your library in time, for example, which is like almost like a false positive in most cases. So by shifting that all the way back to developer, you allow them to shorten the cycle. They don't duplicate the process.

[00:20:09] JM: That security product, is that just a scan using the publicly available vulnerability database or are you guys doing your own security research and finding your own vulnerabilities?

[00:20:22] MJ: So for now, this is all using public accessible data. As far as I know, we are developing our own stuff and we are also now considering moving certain security scanning items down there, meaning instead of having it as part of enterprise edition only, having it as community edition.

[SPONSOR MESSAGE]

[00:20:48] JM: Cruise is a San Francisco based company building a fully electric, self-driving car service. Building self-driving cars is complex involving problems up and down the stack. From hardware to software, from navigation to computer vision. We are at the beginning of the self-driving car industry and Cruise is a leading company in the space.

Join the team at Cruise by going to getcruise.com/careers. That's G-E-T-C-R-U-I-S-E.com/careers. Cruise is a place where you can build on your existing skills while developing new skills and experiences that are pioneering the future of industry. There are opportunities for backend engineers, frontend developers, machine learning programmers and many more positions.

At Cruise you will be surrounded by talented, driven engineers all while helping make cities safer and cleaner. Apply to work at Cruise by going to getcruise.com/careers. That's getcruise.com/careers.

Thank you to Cruise for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:22:08] JM: I just talked to Sneak, they actually have a thing where you can license their vulnerability database. That might be useful for you guys.

[00:22:17] MJ: Right. That might be useful. Yes.

[00:22:20] JM: The GitLab product development process. So I guess you had this sort of random engineer in the company, or presumably not random. I should interview whoever this was at some point. Whoever said, “We’re going to bundle in CICD,” and ended up taking your company to market in a really powerful way. That was one product expansion thing, and then the security product expansion thing happen at some point. Then you got Meltano, which as far as I know is kind of a Sid top-down, woke up one day and said, “We’re going after data engineering.”

There’s a very curious set of product expansions. Is there a standardized process by which new GitLab products get developed, or is it just kind of opportunistic?

[]00:23:06MJ: So our mission is everyone can contributes. That means if you have a brilliant idea, it doesn't really matter which part of the company you come from. Your idea does count. You can propose it whichever way you want. But the only way we actually develop product is using our own products. So the only thing you need to do is propose your idea in our public issue tracker, by the way.

So that means that whoever has had what they think is an interesting idea, they can propose it and they can explain the benefit of us doing that. That gets exposed in various Slack channels or various ways it gets exposed. You can publish your own thing inside of our internal company agenda as like, “Hey, I think I have an interesting idea.”

More often than not, people take in. They try to like poke holes in the idea. They try to like understand the value of it, and the everyone can contribute mantra that we are using doesn't only count as create an issue. You can also develop it right then and there. During the company time, it doesn't matter. You can do a POC to explain like what kind of benefit this brings.

We have that thing for a lot of our features, like package features that we currently have. The container registry came the same way. Someone said like, “Hey, there is a public open source project that does this. How amazing would it be if we – Within our CI, where we already build our code, store those images right away?” They built a quick POC, and that POC blew our mind. We’re like, “This is amazing. So let's do it.”

POC then gets expanded, merged, and we continued iterating on those changes. So there is no real structured way you, need to submit three copies of this, and that you propose your idea. If you're interested in expanding it, you can continue doing it yourself and bringing others to help.

[00:25:05] JM: Is GitLab one big Rrails app?

[00:25:07] MJ: A couple of years ago, yes. I would say that, but no longer. We have a couple of satellites repositories that are communicating with our main Rails app through the API. So we kind of observed the bottlenecks that our application has, specifi cally at on gitlab.com where we have like millions of users. As we observed bottlenecks, we tried to improve the situation obviously, like resolve optimizing the queries and whatever. But then at a certain point, you can only optimize so much, like Ruby has its own advantages and disadvantages and we extract that into a faster environment.

So for example, our now common interface for dealing with Git storage, a project called Gitaly, is completely written in Go. So every time you query for anything on Git data, it goes from the Rails app back to the Go lang library and it serves your quest. We are structuring our obligation that way that as soon as we reach a certain bottleneck, we expand it into a separate service.

[00:26:14] JM: So broadly speaking, GitLab is this like a big monolith and there are some adjacent functionality. Is there a broader lesson we can take from this in this microservices frenzy that we are living through, or is GitLab an exception?

[00:26:35] MJ: I don't believe GitLab is an exception. Again, personal opinion is that you are competing with other projects on your market. So do you want to go and follow a certain philosophy, like microservice all the way that complicates your development workfl ow? You might do a completely perfect microservice-oriented app, right? Like completely cloud native app that will be overtaken by a monolith just because they can develop faster.

What you actually need to care about is the performance of your app, how quickly can you resolve the problems as you see the performance bottlenecks and how quickly can you deliver value to your users and your customers. I think we are doing that actually really well. Obviously, there are all these performance issues, like as soon as you have some codes, that's performance issue right away. But the fact that we tapped into Ruby on Rails engineering pool to develop quickly and then we are now tapping into a Go lang engineering full to increase performance, to improve our app further just kind of shows to me at least that you should be focused on resolving a problem rather than over optimizing your solution.

So could have we chosen a different language to start with? Probably. Would it be better? Maybe. But for us, the fact that we started at that time, Ruby on Rails, and now we are expanding into other languages has been very beneficial for us. That tight coupling between services versioning-wise has actually helped us ensure that we can control things much quicker. That is different for SaaS obviously will. We ship our software. So that is definitely different than if you have a SaaS. If we were primarily and only SaaS company, I think I will be telling you a different story here.

[00:28:29] JM: Assuming some varying levels of experience in our audience, can you define the difference between the way that GitLab is deployed and served and the way that a SaaS product, maybe Zendesk, or Stripe, or something, some SaaS product, what's the difference?

[00:28:51] MJ: So I think the biggest difference for us is that we need to ensure that our customers who are using GitLab don't have to spend time understanding how GitLab internally works. They need to understand how GitLab is used. So if we take our whole application and give it to our customer and say, Well, now you need to update this part of the service. Oh! You're running version 1.1, but you actually need to run version 1.4." You kind of occupy them with a busy work basically, the busywork of I need to maintain your tools.

The way we are doing it, and I honestly believe successfully, is we are taking away that complexity from our self-manage customer, and we are telling them, "Get this big monolithic package. It has everything that you need. Don't worry about the versions. We are taking care of making sure that the versions of the libraries we are using are compatible and they're functional. You just need to make sure that you stay on top of our product development.

So don't lag behind too much. Even if you do lag behind too much, don't lag behind more than a year. So we made that process extremely simple, and with making that process simple, we get our customers to have like the latest features as development, like basically getting them hyped up follow our path and follow our vision.

If we were to do that in a SaaS, no one will actually care how it goes. They would only care how it performs and what kind of interface they have. So what we are actually doing is offering that SaaS story of, “I don’t really care how it works in the background to our self-managed customers,” and that has a lot of value.

[00:30:36] JM: So it's ironic, because if I understand correctly, because a lot of the GitLab customer base is deploying GitLab on premises, you want to give them a monolith, because the monolith is easier for them to deploy, which is very similar to, if I am consuming a SaaS product, I just want to consume the API and I want it to be as smooth an experience as possible. The way that a smooth API-based SaaS experience is made smooth and highly possible is probably through decoupled microservices architecture.

[]00:31:15MJ: Correct.

[00:31:16] JM: Pretty ironic.

[00:31:17] MJ: Yeah. Right?

[00:31:19] JM: That said. So you have this thick, basically monolithic thing, that you’re deploying to customers, you’re delivering to customers. I shouldn’t say deploying. You’re giving it to customers. You’re in charge of the creation and the distribution of updates to GitLab, right?

[00:31:35] MJ: Not anymore actually. That was my job up until January this year. Yeah.

[00:31:40] JM: What do you do now?

[00:31:41] MJ: So I am now responsible for our SaaS offerings.

[00:31:44] JM: Okay.

[00:31:45] MJ: My responsibility now is to take whatever my former team is building and ensuring that that is easily deployable on a SaaS offering the size of gitlab.com. So the delivery team that I'm leading is responsible for release managements. Making sure that those packages actually go out in time for customers and so on, but also for our Kubernetes migrations. So gitlab.com is now running partially on Kubernetes. This is because gitlab.com is really, really big and we need to ensure that we can scale it further.

I also have another team called Scalability, which is responsible for – Do you remember those bottlenecks I was telling you about? So we are observing those bottlenecks on gitlab.com scale. We have our SLA's and SLO's that we are following, and recognizing those bottlenecks as early as possible so that our customers, self-managed customers, never even get to them.

[00:32:41] JM: Nice dog fooding.

[00:32:42] MJ: Exactly.

[00:32:43] JM: So how does the experience of – Actually, let me understand the full lifecycle here. So it sounds like the team that you used to be in charge of, the distribution and delivery release management team for the on-prem product, that product feeds in – So they are basically earlier in the lifecycle than where you sit now, which is the delivery of that product as a SaaS offering.

[00:33:17] MJ: Story is a tiny bit more complex than that, but yes. That is the basic concept. So the distribution team is responsible for building those packages, as in they need to ensure that whatever we are packaging in that moment in time is ready. What my current team is doing is making sure that we are deciding on that timing. We are deciding on when are we going to release to our customers.

So we are both the users and the – I don't even know how to explain it, but we are both the users and controllers of how our software leaves our company. So once we decide that things need to go out, we use the packages and Helm charts that the distribution team producers. We deploy them all automatically obviously to environments. Once we are sufficiently satisfied that everything is okay, we vet and say like, “Yeah, this is ready to go to public.”

[00:34:15] JM: You mentioned Helm charts there. You were largely responsible or played a big role in the move to Kubernetes. Tell me about how GitLab looked before and after the migration to Kubernetes, and I want to understand what value you get from being on Kubernetes.

[00:34:34] MJ: At some point in time, we had – I don't even know how many machines, 200, 300 VMs, which had GitLab deployed on them. So GitLab monolithic package.

[00:34:47] JM: This is GitLab.com, the SaaS?

[00:34:48] MJ: For gitlab.com, specifi cally. Yes. That's meant that every time we hit a certain bottleneck – So, I don't know. For example, Microsoft buys GitHub. Everyone goes to migrates to gitlab.com. So now you have SREs frantically booting up VMs, terraforming stuff and installing things and getting them
online.

[00:35:11] JM: Oh, so this is true story.

[00:35:11] MJ: This is true story. This was like two years ago, I think. 2019 – Yeah, two years ago. As soon as the news hit the press, all SREs came online to start booting up new VMs, and that is – If you consider it now in 2019, that is a kind of bit silly given that you have a platform that does that for you automatically, right?

What we did over the course of this past two years, basically, is we develop those Helm charts that allows to now deploy into Kubernetes. We containerized parts of the application. Well, all of the application, but in different ways. Now, another competitor, can be bots, and we can scale that demand automatically, right? It doesn't really matter.

[00:35:58] JM: Who else is there?

[00:35:59] MJ: I don't know. Yeah. So the difference for us now is that we can actually focus on the application itself rather than what I just said. Again, the background, the infrastructure, like making sure that we can actually continue scaling with further demand quicker than we used to be able to.

[SPONSOR MESSAGE]

[00:36:26] JM: Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:38:16] JM: You and I had a conversation at some past KubCon. I can't remember which one. I can't remember when it was. Time is fading. The conversation essentially centered around the optimal amount of empathy that an engineering manager should have for an engineer. Particularly in – I think we're mostly talking on the context of remote.

I took the opinion that, basically, engineering managers should layout strict requirements for an engineer, and the engineer should go ahead and implement those things. If they can't implement them or they hit bottlenecks, they should communicate very clearly back to the engineering manager. There should be a very clear unemotional base level of interaction between those two people. Beyond that, like, yeah, they should form a friendship. They should be friendly. They should develop water cooler conversation and so on.

But basically, it should be – The backbone of the relationship should be cold, unemotional and productive. I'm not sure how strongly I believe in that, but I kind of wanted to just test you a little bit and have a good conversation. You took a fairly oppositional approach. You're more on the side of high- empathy. Tell me what you think I'm wrong about in that perspective.

[00:39:44] MJ: I don't necessarily think that you're wrong about something. Some people need that approach, the approach that you just said. My experience is majority of the people I worked with react better when they are treated with some empathy, with some understanding, and they actually over-deliver in majority of the cases, because they know that they can – IF the time becomes hard, they know they can count on you to understand them.

So I'm of an opinion that I would rather cut some slack to my engineers that if they missed a deadline, we can sit down and communicate why they missed the deadline as long as they – I don't know if I can say swearwords here. As long as they don't sell me bull. Don' lie to me, and everything will be fine.

If you didn't feel like working that day, just tell me you didn't feel like working that day. I also have those days. I would rather play GTA than work sometimes. But don't tell me like, "Yeah. Well, this was too much of a problem. Too hard." I'm an engineer as well. I know how hard problems are.

Another thing that I also realize, if you gives some space to people, not to lose, but some space to people, most good engineers most of the time find better solutions than you can imagine yourself. I prefer hiring smarter people than I am, because I know how I can work with them and I know how I can get much, much more out of them. Then I can get out of myself. So by serving a tiny bit more empathy and spending a tiny bit more effort in making people appreciate it for what they do, in my opinion, I get like we way better results.

[00:41:32] JM: But by the way, I don't think what I was advocating in that conversation was necessarily a lack of empathy. I was more saying, “Look. Remote management, totally new field.” In my experience, there is so much fluff, and my one other full-time coworker, Erica, is going to listen to this and maybe she'll have her own opinion. We add fluff to our daily conversations. She works in Seattle. I work in San Francisco, and we have our emojis and our hello, how are you doings? Blah-blah-blah.

But ultimately, a lot of it is fluff, and I think over time we've actually gotten less and less fluffy as we have gotten more and more unspoken amount of trust. It's become more and more rudimentary conversation, which is more productive, because fluff is noise. You'd rather have a high-bandwidth, low word count conversation.

So if that's the case and you have an established company like GitLab, why not say that from day one, like, “Look. Empathy is unspoken.” GitLab principle number 15. Empathy is unspoken. Can I make a pull request in this podcast for that to be one of your principles?

[00:42:53] MJ: You can absolutely always submit a request. Absolutely, even if you don’t work for Gitlab. So I think we are mixing lines here. I don't believe in fluff. That's also what I told you a bit earlier. Don't lie to me. Be direct. Tell me what you need and I'll help you out and I'll tell you what I need. So in my opinion, it doesn't have anything to do with empathy. I can be direct while still being empathetic to the person. I can tell them like, “Hey, this needs to be done today. I understand that you have this, this and this, but how can we work this out to mutual agreement?”

I don't have to go and say, “Have you seen the latest episode?” or whatever, and like start by like giving the – That type of fluff, right? That type of inefficient conversation. So I always have efficient conversations with my team members. Everyone knows where they’re standing with me, and when I get upset, this is the difference. I don't yell at them. I just say, “I'm disappointed,” and this directly states how I feel, and I don't have to like go around and say, “Well, you know how you made me feel?” Well, no.” “You should have done your job, or I should have done my job.” You can still have a clear-cut conversation without adding that a fluff that you’re mentioning, right?

I think we are basically in agreement. We are just using a different terminology. I have empathy when I work with my folks, but I also set clear expectations. Whether you want work five hours, one hour, three hours, I don't care. Whether you are in Bahamas that day, I actually don't care. I care as a human being as someone out of the office, right? Like [inaudible 00:44:33] you mentioned. But if we agreed that you're going to deliver this thing until tomorrow, whether you spend four days not doing anything and then hammers everything under the deadline, or you’re going to like do two hours a day and get done, more the power to you.

I have an example to share there, where I got my whole team in one location. We call that fast booth, when you create a new team. Get them all in one location to get to know each other and work on a specific problem. This is exactly where I learned how different my team members are, and with some of them, I worked for years. One guy spends two hours reading stuff on his laptop and wouldn't say Facebook, about Hacker News and knows what. He would be chatting and like not even doing the thing that we said that he needs to do, but then at some point, he just takes the wrap of his laptop and puts his headphones on and spends two hours of the most productive work I’ve her seen in my life and delivers that something under the deadline. So more the power to him. Other people need six hours to achieve the same result.

[00:45:49] JM: What have you changed your mind about in the purview of remote work? What did you do when you started doing remote team management around 2016, I think, that you now do differently? What have you changed your mind about?

[00:46:10] MJ: I think I spend too much time at the beginning of the remote work when I build a team in – Now I call it micromanaging. Then I called a project managing. Making sure that people committed their code, pushed their code, made their code visible so that I can check it out and understand where they are.

Obviously, some of that comes with seniority of the people that you have working for you. But now I'm more of an opinion, deadlines, expectations, clear expectations, clear agreements are more important than me seeing that code right the very moment you wrote it. This point in time, I just only care that if we agreed that this is doable in this time, you are delivering that. If you're not, if you notice that something is going off, your communication style needs to be direct and like you need to raise urgency right away. Excuses like, "Oh, well. This took more time, because I didn't anticipate X, Y, and Z." That do not matter to me if you didn't raise it up or expose it in some way.

That is kind of the opposite of what I experienced in the office is where I always had a feeling that my boss was looking over my shoulder to ensure that I deliver some results. It's the worst feeling in the world, someone looking over your shoulder.

So now I actually have conversations on why did you miss the deadline? Where did we miss our mark on estimating these things? How can we ensure that this never ever happens again? Then, again, like I said, if you want to be in Bahamas for a week, be there. I don't mind. Just deliver it as we agreed on that. I think that was the biggest, biggest switch for me.

Now, I actually look at codes as, well, out of hobby basically, right? When my engineers deliver something, I'm curious. I want to see like whether I could have figured out a more optimal solution, and sometimes offer the more optimal solution. But when they did it, that's no longer my issue.

[00:48:08] JM: GitLab has these set of unique traits, these set of unique engineering practices and cultural practices. It's a remote work organization. Sid it is a very distinctive CEO. I think when you have a company that is that much of a maverick, in some ways, people are waiting for something to go wrong with the company. There are some people who are almost rooting against it. They want the norm of engineering and business to remain the same for whatever reason.

So there were a couple of recent events around GitLab that cast the company in a negative light. There were some "controversy". I didn't follow any of it that closely, but maybe you could give me your perspective from the inside. How did that controversy feel? What were the controversial events, and what was the outcome?

[00:49:16] MJ: So we had a couple of events. One was the telemetry challenge, I can call it. The other one was data residency issue, meaning our proposed moratorium on hiring from certain countries. Both of those came – I don't know how to explain. It is interesting, because all of those items were public for months. They were out there for anyone to pick it up while things were being discussed, including the telemetry update.

[00:49:46] JM: What was the telemetry update?

[00:49:48] MJ: Telemetry update, this is where we made a mistake. We send the email with an advanced – Not warning, but advanced information that we are planning to track behaviors on gitlab.com, so our SaaS offering. Due to unfortunate set of events, the email that was sent out was not really super clear what that actually means for our self-managed customers.

So in certain ways, you can interpret it as all of GitLab is going to now be tracked. The other one, the data residency issue, certain customers of gitlab.com have a valid request to know who has access to their data. Due to various reasons, there was a requirement that any Russian resident and Chinese resident is reported as having access to their data.

I mean, I honestly think that's a valid question. Now, what we did about that, that's a different story. So we had a discussion in a public issue between the highest level executives on how can this be done and what are the reasons for this being done. Unfortunately, whether people did not understand how our company works. it doesn't really matter, because they understood our discussion as final, like a thing being done. As in GitLab decided to do this, which is not how we actually operate. We operate in public, and until we find a good solution, we continue discussing it, right?

I think what was an unfortunate outcome at that moment in time while it was happening is that we are now a big company. We also have a lot of people inside of the company who are very opinionated about some of these situations, right? When news hit Hacker News, instead of assuming positive intent, which is one part of our values, we had a situation where some lack of trust was exposed towards the decision-makers.

So what I think was brilliant was the result or rather how we handled what was happening at that time. So this is where I really love working for this company. We recognized the mistake. We recognized immediately that as soon as this went out and people start raising their concerns about it, we understood where we made our mistake and we sent out a follow-up saying, “You're right. We made a mistake. We didn't understand that you can – We were to set on what we understood was the problem and we didn't understand the optics of how this looks to someone who is outside of this whole story. That's one thing.

Another thing I think that was really interesting was how internally all of that looked. The fact that we have so many people in the company that care about the company and the mission of the company is really rewarding, because everyone started offering solutions, “Hey, we can do this. We can try this way. Don't forget about the open source. We need to communicate this with the rest of the open source community. How are we going to do that?”

Everyone kind of rallied around, “Here are the solutions to the problems we created for ourselves.” Actually, this morning, we had a big meeting – It’s not a meeting, but I was the moderator together with a couple of other colleagues in a conversation about how do we align our decisions with our values, where equal participants were the executives. So ET members and any team member that wanted to participate in this discussion.

So what actually happened was people asked really probing questions and are executives laid out, “This is the thing that we are thinking about right now, and these are the reasons why we are doing that. This is a business requirement we have. This is what board set for us, and this is how this affects our numbers.”

So whether that was a good decision or not. Now, hindsight, we know some of these items could have been handled better. But what was important in this whole conversation was that people that contributed now understood the full background of this situation, and we also found new alleys of communication with, well, not only executives, but like both sides. How are we going to raise up these certain things in the future in a more constructive way?

Again, assume positive intent. I'm actually excited to see that there is a second side to this controversy, and that was humans behind it actually behaved like humans, like these people, where they actually started discussing things and understanding the background from both sides. Not only executives sharing with team members, but team members sharing with executives.

[00:54:43] JM: It's such a commentary on the modern problems of Internet communication, where people do not assume good intent in modern Internet communication. People assume the worst intent. There are many people who will – They would rather have a subset of the available information that lets them assume bad intent then consume the suffi cient quantity of information to understand the full story when the full story will reveal a more balanced, and humane, and positive intent.

So it's just funny, you hear plenty of discussions around social media, public discourse in social media and Facebook or twitter or news commentary or hacker news or Redit or whatever. It's just funny kind of seeing this also be true for company that operates in the open, almost the GitLab social network or the GitLab open operating doctrine.

So I'm sure we could talk that a lore more, but we're almost out of time. I mean, that could be its own show. Maybe I can get Sid to come back on. We're at KubeCon. Are you going to Reinvent?

[00:55:56] MJ: No, I am not.

[00:55:57] JM: You are not going to Reinvent. Have you been before?

[00:55:58] MJ: No. I have not.

[00:55:59] JM: You've never been. Okay.

[00:56:00] MJ: It's in Las Vegas, and Las Vegas is tricky.

[00:56:05] JM: Not the biggest fan? Okay. Yes, tricky. Why is this conference relevant to you?

[00:56:09] MJ: I think one of the most – The biggest reason why this is relevant to us is we do offer tools that enable folks to either deploy to Kubernetes or work with Kubernetes in some shape or form, and personally as well for me as a the leader of the team that is handling gitLab.com on Kubernetes, learning the new practices are the best practices or whatever this community brings. The community that is growing so fast that it is not believable, honestly.

It is really important to stay on top of those trends, because there are a lot of things that are still not resolved in the platforms, right? State is still a tricky subject, and we don't have states in Kubernetes workflows, right? How you deploy to Kubernetes is also like an interesting topic that I have followed a lot of topics this time around where a lot of folks are having the same problem. Interestingly enough, they're creating their own tools to create the same workflow, and I'm here to tell them that, "Yeah, GitLab offers already all of that for them.

So they don't have to." But it is really, really important to stay on top of those trends, because the cost of the infrastructure, the cost of people handling that infrastructure is continuing to rise with the demand of the services that you have to offer. So anyway you can make things easier and more cost efficient for yourself will give you the competitive advantage.

[00:57:40] JM: Marin, thanks for coming on the show. It'd be a pleasure to have you back anytime.

[00:57:44] MJ: Always.

[00:57:45] JM: Okay. Thank you.

[00:57:46] MJ: Thank you.

[END OF INTERVIEW]

[00:57:55] JM: As a programmer, you think an object. With MongoDB, so does your database. MongoDB is the most popular document-based database built for modern application developers and the cloud area. Millions of developers use MongoDB to power the world's most innovative products and services, from crypto currency, to online gaming, IoT and more. Try Mongo DB today with Atlas, the global cloud database service that runs on AWS, Azure and Google Cloud. Configure, deploy and connect to your database in just a few minutes. Check it out at mongodb.com/atlas. That's mongodb.com/atlas.

Thank you to MongoDB for being a sponsor of Software Engineering Daily.

[END]