EPISODE 791

[INTRODUCTION]

[00:00:00] JM: Gitlab is an open source platform for software development. Gitlab started with the ability to manage git repositories and now has functionality for collaboration, issue tracking, continuous integration, logging and tracing. Gitlab's core business is selling to enterprises who want a self-hosted git installation, such as banks or other companies who prefer not to use a git service in the cloud. The vision for Gitlab is to provide a platform for managing the full software development lifecycle, from code hosting, to deployment, as well as tools for observability and project management.

Sid Sijbrandij is the CEO of Gitlab and he joins the show to talk about the product, the business and the company's ambitious vision for the future. Gitlab's strategy is to offer a set of tools that work for developers out of the box, cutting down on time spent integrating each individual vendor. It's really an ambitious vision, but it actually makes a lot of sense if you hear Sid explain it, and it was really fun talking to him in this interview. I enjoyed it quite a lot. I want to thank the Linux Foundation for inviting me to the open source leadership summit, which is where I spoke with Sid. That was a lot of fun.

Before we get started I want to mention two events that we are having in-person. On April, we are having a Software Engineering Daily meet up with Haseeb Qureshi, a frequently requested guest on the show. Haseeb is a speaker who can talk about career development, software engineering, cryptocurrency investing. He was previously an engineer at Airbnb and is now an investor. So that's at softwareengineeringdaily.com/meetup.

The other event is the hackathon that we're putting on. You can go to softwareengineeringdaily.com/hackathon. That's April 6th at App Academy in San Francisco, and you should sign up now because we're going to have limited seating for both of those events. The hackathon for anyone who's working on an open source project, a business, an artistic project that is somewhat technical, and we're going to be using the platform that I'm building called FindCollabs, which is a platform for finding collaborators for your projects and building new things. So if you want to check that out, you can go to softwareengineeringdaily.com/

hackathon where you can see the product that I'm referring to called FindCollabs at findcollabs.com.

The hackathon has a $5,000 price purse. So there is a competition element to it and some money to be won, but it will also be a ton of fun. So I hope to see you check it out. You can go to softwareengineeringdaily.com/meetup for that meet up details and softwareengineeringdaily.com/hackathon to find out more about our hackathon.

Thanks for listening and I hope you'll enjoy this show.

[SPONSOR MESSAGE]

[00:03:18] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his

interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW]

[00:05:25] JM: Sid Sijbrandij, welcome to Software Engineering Daily.

[00:05:27] SS: Thanks for having me.

[00:05:28] JM: Before we discuss Gitlab, I want to start with the topic of git itself. So git was initially just this version control system built for Linux, but it's evolved to become a tool that's foundational for software engineering. Why has git had such a big impact on the world of software?

[00:05:46] SS: I think because it was the first distributed version control system, or almost it shares that [inaudible 00:05:53] that was open source. When people hear distributed, maybe it doesn't resonate why that's important. I think that the reason that's important is, before, you could fork off a project, but you can never bring them back together. Because git was made from the ground up to be forked off and be merged back together, it became much easier to collaborate.

Before, if you wanted to collaborate, you had to get invited. So it was like, "Okay, I want to contribute to your software. Can I please have access on your SVN server?" Then there was – They said, "Well, are you good enough?" etc. There's so much friction. With git it was like as soon as you can read the software, you can make an improvement to it and only then people have to decide whether they want to incorporate that.

So from having to be invited to the party, to everyone is welcome, and I think that's why a revolutionized software development, and now even infrastructure, you have people saying that git ops version control, like your changes in infrastructure is one of the best things that happened, and I think it's going to revolutionize more, book writing.

So we see a future where all digital cultural goes from read only to read/write. When you now view a movie, you get a binary, you get the end results. Why not give people access to all the raw camera feeds and all the editing information so people can improve upon it?

[00:07:23] JM: You think git is medium for that.

[00:07:25] SS: For now it's looking like it has a lot of momentum and there's a lot of companies investing and making sure that git can ever grow more capable and deal with large files and a lot of files. So I think it has a good shot at it.

[00:07:38] JM: So you think git could potentially be the underlying medium for managing art projects, and design specs, and blue prints, and music files.

[00:07:48] SS: I do think so.

[00:07:51] JM: What are the different business models that people had built around git?

[00:07:54] SS: So there are different things I think. We have a famous competitor called GitHub, and they said, "Look, git is great, but it's hard to use. So we'll make an easier interface on top of it." Everything they built on top of git – Or not everything, but to a large extent, that was proprietary. We came after them and we said, "Even the stuff we built on top of it to a huge extent, that's going to be open." So a more open core business model. Of course, there's also a lot of companies that just use git as a part of their day-to-day activities and they're not known for that, but it's helping them be more effective.

[00:08:33] JM: Gitlab's core business is the self-managed git. Why is the market for self-managed git increasing?

[00:08:43] SS: Yeah, I was kind of puzzled by that. I started with gitlab.com and I thought the SaaS is the future that will make all the money. Today we have a great SaaS offering, gitlab.com. It's used by millions of users, but a lot of companies, especially the larger ones, they prefer to run their own installation of Gitlab, and we made that super simple. So maybe that's

one of the reasons it's popular. Also, where your data is getting much more important and integrating with other tools tends to be easier if you run it for yourself.

[00:09:18] JM: Many of the older organizations that are adapting git today, they still have older version control systems in place and they're going through this migration path of moving from one version control system to another. What does the adaption path look like for an older company, like a bank that's migrating to git today?

[00:09:42] SS: Yeah, most companies have things like clear case, or Perforce in them, and most of them when they adapt Gitlab, they stand up one of their newer teams with git and then overtime they invite other teams in their company convert and when the time has come for them, they offer their code to the new format and they start off. This has been easier than many people anticipate. Most of the time the adaption of Gitlab goes twice as fast as the company had anticipated. So we tell them, "It's pretty easy. Your engineers will love it and they will adapt it," and they say, "Well, that might be true, but we've never implemented something faster than X, Y, Z, and six months later we'll double the rate."

[00:10:31] JM: So Github is a social network built on git, and Gitlab also has social elements. Describe how individuals communicate on Gitlab.

[00:10:42] SS: Yeah, I think Gitlab in contrast to Github is more commonly used with a team of coworkers. So to simplify it a bit, where Github is close source software that's used to develop open source software. Gitlab is to a huge part open source software that is used to make closed source software. So most of the people using Gitlab use it in their day job. They may be use it as a hobby, but where Github is used for the majority of open source, Gitlab is used by the majority of enterprise companies to develop software.

[00:11:18] JM: For many years, Github was by far the biggest company that have been built around git. Why did it take so long for another business model around git to be found?

[00:11:33] SS: You have to ask my cofounder, Dmitriy, why he started this only in 2011. That was the year that he needed something like Github, and his company wouldn't pay for it, and he couldn't pay for it either. He was living in Ukraine. Every morning he went to a well to get water.

He came home with a few buckets, but he was very determined to have great collaboration software and he set off and said, “I can make it myself,” and a year later 300 people joined him.

[00:12:05] JM: So as we’ve said, Gitlab’s primary customer base is these on-prem installations. How does the usage of on-prem git differ from somebody who’s using git as a SaaS?

[00:12:20] SS: I think there’s not a big difference with Gitlab. Even the on-prem installations, most of them run in the cloud. It’s the same software. If you use gitlab.com or self-managing sense, it’s the exact same software you’re running. I think the difference is on the SaaS, there tends to be a million users on the installation, but a self-managed one, it’s just the people like your company and some suppliers. So it tends to be in the thousands, although there are companies running installations with more than 30,000 people.

[00:12:49] JM: Do customers pay you just for the initial integration, or is there also ongoing support?

[00:12:55] SS: Yeah, the majority of our income is the subscription. So you pay per user per year.

[00:13:01] JM: Okay, and they’re paying for the on-prem version plus some additional features?

[00:13:11] SS: Pricing is the same between the on-prem or the self-managed version and .com, and you pay for extra features and support. Majority of what people pay for is the extra features. So we have three paid plans, and depending on the plan, you get more proprietary features. Those features are aimed at leadership in the organization. So if something is mainly required by individual contributors, it’s open source. If something is mainly requested by executives, it’s in our highest paid plan.

[00:13:45] JM: What are some of those executive features that people want?

[00:13:48] SS: For example, a roadmap of all the company initiatives, a value stream management where you can see how you – What’s your release cycle looks like and a security overview throughout the company of any vulnerabilities that exists.

[00:14:04] JM: And what's the typical sales process like for Gitlab? If a large enterprise starts using it and maybe they start using the free version, they end up making a purchase. What is that sales roadmap look like?

[00:14:19] SS: Yeah, basically we have four different types of buyers. We have the people that don't buy, the individual contributors. That's like 90% of the users of Gitlab are on the free version. Then we have small and medium business, and it's mostly self-serve. So they go to our website and they purchase a subscription there. The midmarket, and we call that sales assisted, or inside sales. So they can reach out to sales people, but in the end they are directed to our sales service site to make the actual purchase. At the top of the market, that's enterprise sales. That's strategic account leaders, solution architect, visit the customer, and those tend to be the larger deals.

[00:15:01] JM: Now, from a product standpoint, what I find really interesting about Gitlab is the vision that you've laid out to go after the full stack of software delivery. I think to some people listening, it might to be intuitive why Git is potentially a foundation for going after the full stack of the software lifecycle. But I think continuous delivery, we could use that as a good example. We can go through some other examples. Why is continuous delivery closely tied to a user's git workflow?

[00:15:44] SS: Yeah. I think what you're seeing is that devops is getting more complex. There are things to do, and people struggle integrating all the different tools that they need for that. The typical enterprise company we encountered is between 50 and 100 people working on integrating all those tools together. That's differentiated heavy lifting. They could share that burden with other companies.

So what Gitlab has evolved from just being version control to being 10 times more. So you can manage across the organization. You can plan what you're going to build. You can create that with a web ID in Gitlab. You can verify that your code is performant and that the test passed. You can package that all up. You can deploy that. You can configure that. You can monitor that and make sure it's secured and defended against attacks. So that's a way bigger scope, and we

think that's way better in a single application and that's not something that we decided, but it's something we learned.

We had an engineer from Poland called Kami, and he made a better version for Gitlab CI, our continuous integration platform, and we embraced that and we said, "Okay, from now on your version is the official version. By the way, do you want to come work with us?" and he did that. A bit later he came to Dmitriy, my cofounder. He said, "Hey, we got to bring these two projects we had, Gitlab version control and Gitlab CI. We're going to make them into one application." Dmitriy said, "You're obviously wrong. That's not how anybody else does it. We shouldn't make this monolith of an application. It will be disastrous for our code quality and for the user experience."

Then then they came to me after Dmitriy was convinced and he said, "You are obviously wrong. We need to have tools that are composable integrate with other tools. That's the Unix philosophy." He said, "Well, if you don't believe that it's better for a user, at least believe it's more efficient for us, because we only have to release one application instead of two. Efficiency is in our value," so he was right. So I said, "Okay. Well, that makes sense. I'm no longer stand in your way."

He did it and he was right beyond what he even expected. It was so much better experience to have the CI with the conversion control. If you're viewing like a commit to see, "Hey, here's the test that we run on that commit," and to be able to switch that without like going to a new browser window and vice versa. We realized we stumbled on a big secret, a secret now because we're not telling anybody, but because nobody believed us. We doubled down on this philosophy and we started doing continuous delivery and many other things.

Because of that, we're now able to do things that no one else can do. For example, we're going to launch auto remediation this year, and it means that if you run software production and a new vulnerability comes out, say, a heart bleed or something like that, Gitlab is going to automatically detect which software is vulnerable. It's going to fix that. Going to make a merge request for that, going to merge that, sends it out with progressive delivery, monitors whether it's going okay. If it doesn't go okay, it's reverted and it opens up an issue for you to solve. So that when

you arrive at work, you don't have 100 things you need to patch. You have that one thing that wasn't patched that was patched, but showed that it no longer met the service level objective.

That is something that is really hard to build if you don't have everything in a single application, because otherwise you have different definitions. You don't have the right monitoring data, etc. Everything needs to come together, and these are things that the more we integrate, the more value we can create for people.

[SPONSOR MESSAGE]

[00:19:48] JM: How do you know what it's like to use your product? You're the creator of your product. So it's very hard to put yourself in the shoes of the average user. You can talk to your users. You can also mine and analyze data, but really understanding that experience is hard. Trying to put yourself in the shoes of your user is hard.

FullStory allows you to record and reproduce real user experiences on your site. You can finally know your user's experience by seeing what they see on their side of the screen. FullStory is instant replay for your website. It's the power to support customers without the back and forth, to troubleshoot bugs in your software without guessing. It allows you drive product engagement by seeing literally what works and what doesn't for actual users on your site.

FullStory is offering a free one month trial at fullstory.com/sedaily for Software Engineering Daily listeners. This free trial doubles the regular 14-day trial available from fullstory.com. Go to fullstory.com/sedaily to get this one month trial and it allows you to test the search and session replay rom FullStory. You can also try out FullStory's mini integrations with Gira, Bugsnag, Trello, Intercom. It's a fully integrated system. FullStory's value will become clear the second that you find a user who failed to convert because of some obscure bug. You'll be able to see precisely what errors occurred as well as the stack traces, the browser configurations, the geo, the IP, other useful details that are necessary not only to fix the bug, but to scope how many other people were impacted by that bug.

Get to know your users with FullStory. Go to fullstory.com/sedaily to activate your free one month trial. Thank you to FullStory.

[INTERVIEW CONTINUED]

[00:22:10] JM: Okay. So I completely agree with that argument to a certain extent. So continuous integration, yes, closely tied to your git workflow, absolutely. Having that integrated sounds great to me, and you can even take that one step further perhaps, what you're saying with the kind of static code analysis testing of vulnerability code testing. This is stuff that you want tightly either in your git workflow or in your continuous integration workflow, but it's a slippery slope, right? If you go from there to monitoring, and logging, and Slack type communication, you could eventually say, "Yeah, we actually need all of these stuff in one application." Why is that not a slippery slope?

[00:23:06] SS: It is a slippery slope. Now, I believe this slippery slope argument per se. It's not a valid criticism. We should be objective about every single increment and not say, "Well, there's going to be increments in the future that don't work."

[00:23:21] JM: Like it doesn't become Fortnite eventually, or maybe it does.

[00:23:26] SS: I want to object to the commercial success of Fortnite, but we're not selling skins yet, or special packages like that. I do think Slack is a good one, because we seriously considered going to that business, and where bundling matter most with Gitlab and open source Slack alternative. That's a great software, but we decided not to pursue that, because we see in our organization what you want, if you want the entire organization on one kind of chat platform, it needs to be extremely usable, the user friendliness is paramount and interfaced. There's not a big advantage to being able to turn out a lot of features.

So we decided that that is not a core business for us. That's basically where we stop. All the other things are great, and I think you might see that in monitoring. We used to have like logging separate from metrics and that's separate from tracing. What you're now seeing is that the combination is way better. I think Datadog did an awesome job allowing me to go from metrics to logging.

Right now a lot of monitoring companies are working on the trifecta, being able to see an error in metrics, then diving into tracing to see where that slowness is coming from, and then going from tracing into the logs to see the exact error message. So integrated monitoring is better, and that doesn't mean that it should be part of Gitlab, but we think it should be and it currently is. We have all these things in Gitlab, and that allows us to not deploy into a black hole. How can you do continuous delivery without knowing whether you delivered correctly?

Right now I see organizations and they're babysitting their deployment. It's like you have to say, "Oh, put out my deployment and then I'm going to watch the grass for half an hour." That doesn't make sense. A computer could watch those. A computer can be fast in reverting it if it's not looking good, and a human can be. We should have the humans do something more useful and have the progressive delivery being handled by Gitlab. You set your object, like I want the error rate of this, I want the response time of this. If it's looking worst, just revert it and try to fix it before you try again. So I do think monitoring integrated with the rest is a big advantage.

[00:25:48] JM: So you have monitoring integrated with the whole Gitlab process, but that doesn't mean that you throw out your Datadog you're saying, because you have Datadog for doing other things than the git workflow.

[00:26:01] SS: Well, we don't require you to throw anything out. So we work find with Datadog, with New Relic, but also with Jenkins, with Github, with Jira. We don't want to be a close platform. So embrace parts of Gitlab, because you like them, not because you're forced to.

We do think it's great to offer kind of an out of the box that works. If you install Gitlab, you get Prometheus with it automatically. We think that's the future of metrics and want to make sure that's set up. As people develop more and more microservices, you don't want to go start a new microservice and then spend half a day making sure error tracking is set up, and internet management is set up, and metrics, and logging, and tracing. That's should just work out of the box. The only thing you should have to do is install Gitlab and start a new project, and we want to take care of the rest. If you want to go use something else, that's totally cool, but I don't think that's an excuse for not having everything working out of the box.

[00:27:00] JM: So a lot of companies have some kind of end-by-end integration challenge. So a lot of companies that are logging providers, or ops providers – I'm thinking of marketing pages that I've seen where it says, "Works with Slack, works with Datadog, works with AWS," and depending on what company it is, they might have to write the integrations or the other company might have to write the integrations.

It sounds like, from your perspective, since you're open source, you kind of let other people write integrations and you just solve for the out of the box Gitlab experience. Is that right?

[00:27:46] SS: Exactly. So we do have hundreds of integrations, and luckily we have millions of users. If they need an integration, they will contribute it to Gitlab. So that's really helping. I think what you can do if you're in a single data store, because Gitlab is a single application. There's a single data store, a single concept of what an environment is, who a user is, what the writes are, what the application is. Because of that, we can integrate much more deeply and you'd be able to do two APIs. I was skeptical of this. Like Gitlab version control and Gitlab CI were custom made for each other. We had like custom APIs in either of them to fit together perfectly. Still, when it's a single application, it becomes so much easier. Sometimes it feels a bit like we're cheating, because when we make a new functionality, like we recently added feature flags, we don't have to integrate with the world. We know exactly where the application is and what its URL is and what environments it's deployed on. Because Gitlab already did that. So it's much easier for the Gitlab community to add a new feature to Gitlab than it would be to make that a standalone company.

[00:28:54] JM: So once you write feature flagging as an example, you write your feature flagging for Gitlab product and somebody else says, "I'm going to stick with the feature flag vendor that I already have."

[00:29:06] SS: Yes. So they should probably use LaunchDarkly. That's a really good feature flagging tool, and that's great. They can just use that. Using something that's not Gitlab is not any harder than it was.

[00:29:16] JM: Okay. So there's not really like an integration that needs to be written for a LaunchDarkly plus Gitlab?

[00:29:24] SS: It works the same whether you use Gitlab or you use Github or Bitbucket. The handy integration with a third-party works the same. If you use something external, it's just as hard as before. If you use Gitlab, it's way easier than anything else.

[00:29:43] JM: So this is interesting. I have talked to a few people about the Gitlab strategy. So the product vision is this ability to offer a highly integrated, easy to use platform for your full software development lifecycle, and people who have been in the software industry for a while, they look at that and they're like, they're biting off more than they can chew. That's really intimidating. But talking to you, it actually doesn't sound that intimidating. It sounds more like you're just trying to have nice defaults and allow people to opt in to other vendors if they want to.

[00:30:25] SS: That's totally true, and we have a company strategy that says ref over debt. So we want to, as a company, we're continuously pushing out what Gitlab can do. Now, we start off really simple. Our tracing support right now, it still depends on like a Jaeger thing which you have easy access to, but it's not as integrated as we would like. But it takes that – We launched it last year and it's going to take time to mature.

Things that we launched in 2011, 2012, planning, creation, verify, those things are best in class right now. They're not best in class because we have the most engineers. They're best in class because we get the biggest community.

[00:31:05] JM: Is it planning, creation, verify?

[00:31:07] SS: Yeah. So that's like an issue tracker, that's like the git repositories and the CI. For example, Gitlab CI was chosen by Forester to be like the best in the business. It's better than any other alternative out there. Even Pure Place, and that's because people contributed back to it. We have every month more than 100 community contributions coming in of people that use this software.

So if we wanted to make everything with just Gitlab team members, yes, the strategy would work, because you're missing – Like you have everything, but everything is not really great to

use. It doesn't have all the features you need. But because people add the things they're missing, it gets better overtime. Then after 5 to 7 years, it is the best in the business.

[00:32:01] JM: Has that strategy made the surface area of the product any harder to manage?

[00:32:07] SS: It's been remarkable, that although we greatly like expanded the scope of the product, I think the user interface has gone better overtime instead of worse. So far, the technical complexity is also manageable. We expected that our development would slow down and all those other things that can happen when a project reaches a certain scale. I think we have a pretty high-quality codebase. We've chosen a good data model and people are able to add things without other things falling over. So, so far so good.

[00:32:40] JM: So when you introduce something like tracing – So you have distributed tracing in Gitlab?

[00:32:47] SS: Yeah, very rudimentary implementation based on Jaeger.

[00:32:52] JM: Just to give people more context for how that works, like what am I embedding in my code, or what kind of imports am I making, and does this only work during deployments, or is this just like constant monitoring? What does it look like?

[00:33:08] SS: Yeah, it's constant monitoring, and I'm not an expert, but as far as I'm aware, you have to embed to a Jaeger agent in your application. We have a technology called Auto DevOps, which means you just push your code and then Gitlab figures out how to build it, how to test it, how to deploy it and we're also looking forward to kind of automatically embedding these agents so that that just works out of the box.

[00:33:35] JM: So do you have a feeling at all that you're competing with the other providers, like the monitoring providers, or would you not consider yourself competitive with them until you have what we feel like a first-class offering within a given product category.

[00:33:56] SS: Yeah. I think it's too early to call ourselves a competitor. I'm sure that Datadog has never lost a deal to people that said, "I'm going to use Gitlab monitoring." Because it's

rudimentary. We started it like last year. It's so common there. But it's getting better and we'll soon be able to use it ourselves and then we'll start seeing some users starting to use and then we'll start seeing new contributions, and that we need those for multiple years before it's able to compete in the marketplace.

[00:34:28] JM: Who do the contributions to the open source project come from? What is the makeup of the community?

[00:34:35] SS: It's very diverse as you might guess. It's mostly individual contributors at organizations. One very great development we're seeing is that big companies that are buying Gitlab so that they don't have to have like 100 people working on their dev tooling integration anymore, they say, "Look, 90 of these people, they can work on something else, something that directly ads value for a business. But 10% of the people will keep, because if there's a feature in Gitlab that we're missing, we want to be able to add that within a month." So that gives Gitlab even more contributions. So we're really encouraged by that development.

[00:35:17] JM: So if I'm a company like a bank, an old, large bank, and I'm adapting Gitlab. The way that I start to adapt the fully integrated solution is that some individual small team within my organization is going to kind of a be trial user for it?

[00:35:40] SS: That's what we see most of the time. Most of the time we come in and it turns out that there's already teams using Gitlab. There's no big organization that doesn't have some Gitlab servers running somewhere. Maybe the management isn't aware, but they're already using it. Then we come in and we talk to the executives in the company and say, "Okay, let's do a proof of concept. We'll give our most cloud native, our most advanced teams, we'll give them Gitlab, because they need to be satisfied with what it has to offer."

I heard here yesterday that like most of the people deploying to Kubernetes are using Gitlab right now. So every single time, that team loves to use Gitlab. So the proof of concept is a success. Then they start embracing it. For example, Goldman Sachs, one of the most advanced financial services institutions in the world they are based in Gitlab. They thought it was so good, not only did they adapt it, they even said, "Look, we want to invest in your company. Off our own balance sheet," where they participated in our series D, because they see that there are also a

lot of problems that they were having before. They can get their cycle time-to-time between deciding to do something and getting the code out again and get it down from weeks to minutes.

[00:36:58] JM: Tell me more about the struggles that they were having.

[00:37:01] SS: The struggle is, for example, everything that goes out in a big bank has to be security tested. So you need to set a code analysis, a dynamic code analysis, the container scanning, the dependency scanning. That used to happen when they were completely done with everything. When it was time to go live with something, then they those scans and they found stuff. If you can do that earlier, you save a lot of time.

With Gitlab, as soon as people push their code, those scans run. Even if it's not ready for production yet but they get the feedback immediately, the security team is able to see those things. They'll show up in their dashboards. So just earlier, that all helps to compress the time it takes between making something and deploying it and monitoring what it does.

[00:37:46] JM: At Goldman Sachs, they're still just very newer teams that are using or has it propagated throughout the organization?

[00:37:52] SS: You know it's propagating. We start with the newer teams, because they tend to be the fault leaders in the organization, and then the rest of the organization joins.

[00:38:02] JM: So earlier we were talking about this idea of git potentially being an underpinning for managing all these other kinds of file types. Do you have a vision for how that might unfold, or is that just too far in the future to actually think about?

[00:38:26] SS: It is part of our mission and our vision to do this, to change culture from read only to read/write. In our seed round for example, Ashton Kutcher participated, because he's seeing the same thing, that things should be open to contributions. Things like movies, but everything we interact with. I hope that one day in the future when I use a chair and I'm happy with the chair, I can access its schematics and like suggest an improvement to it.

For now, we're seeing that the next thing after code is probably going to be websites, so static websites. We made a thing called Gitlab Pages that allows you to host the static website. One of the great things was someone contributed recently something that you can make private websites at that. So you have like access control on it. So it doesn't necessarily have to be a public website. Pretty complex to make, not something we were planning to invest time, but someone needed that and they did that.

So after websites, possibly books. There's a great service called Gitbook, which is right now if you read a book and you spot an error, you have to email the author just to say like, "On this page of this edition of the book, on this sentence, this letter." It doesn't make any sense. Why can't I just make the correction and just submit it? They can see immediately whether it's good or not.

So I think books will be another thing. It's hard to predict what's next. I'm very excited about the data science lifecycle. We got another project called [inaudible 00:40:02], completely separate from Gitlab, but we want to do the complete lifecycle from kind of getting data from an API or the way to getting it in dashboard, making sure that's all version controlled, that you know not only what the code is you use, but also the modeling parameters were and where you got the data, what time, data providence, data governance. Then ML and AI are also really exciting, because right now a lot of that is not version controlled, not governed correctly, but it's super hard to do and I look forward to kind of innovating in that space.

But the nice thing is we don't have to do everything. People take Gitlab and they use it for other purposes. For example, O'Reilly Media, probably well-known in your audience. They use Gitlab on the backend to write their books and they wrote their own custom frontend on it for their offers. But on the backend it's Gitlab.

[00:40:56] JM: Oh! They would open source that frontend.

[00:40:58] SS: They have.

[00:40:59] JM: Oh, it is open source. Okay. So I can write a book on top of git.

[00:41:05] SS: You can. No one is hosting their software yet. It's pretty specific to them. I think it's great that they open sourced it. It's called Atlas. You can find it on partner's page, but I think that people of Gitbook, although they're not using Gitlab, I think they're doing a really good job and making it easy to write a book with git. So that might be something you consider. Otherwise, publish a static website and use Gitlab Pages.

[SPONSOR MESSAGE]

[00:41:36] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[00:43:00] JM: So you just casually mentioned you're going to go after the data providence, data science, AI, ML market. I mean, that's an example of a category where I did a show with Pachyderm, like three or four weeks ago. He's been working on Pachyderm for like four years, and there's still just scratching the surface of the problem and they're very happy with where they're at. But I just use that as an example of – I mean, it's both why I like Gitlab. You're an interesting company, but it's an ambitious thing to just kind of casually throw out there. Do you feel like your resources are pulled in too many directions right now? Is it a strain at all or do you

feel like you're managing in a way where it's like, "Yeah, we're just making a little bit of progress on that."

[00:43:55] SS: First of all, I think Pachyderm is doing an awesome job and I think they're one of the few companies that has a shot at like version controlling data. Git is useful for version controlling code. They can't version control data, and it's hard to kind of know. Yeah, run this analysis, but what were the exact versions? I know [inaudible 00:44:13] from Y Combinator, we were in the same batch.

[00:44:17] JM: Oh! Okay.

[00:44:17] SS: I know how hard the problem is. I know how good they are on the Super Nintendo. It totally destroyed me. But it's a super hard problem. We're not going after that problem, because I know how hard that is. We're going after a simpler problem where you just have an API, for example, Google Analytics, or Sales Force where you have data in that and you want to make a custom report. To do that, you have to extract that, load it in a database, apply some transformations, make a model of what you want to see, and they say, "Oh! This is our definition of revenue." Then make sure that that gets outputted, displayed. You can use Jupyter notebooks with it and you orchestrate this transformation to happen on a regular basis. That's a more simple problem and that's one we're solving with Meltano, and that's a team of seven people. We just had Daniel Morrill join us, and he's doing an awesome job of leading that team. I think they're already doing things that would take you many, many hours with other tools, and with Meltano you can do it in minutes. There's still a lot of work to be done. It's very rough around the edges right now, but they're going to fix that in a matter of weeks. We have pretty high hopes, because we don't think it makes sense to use seven different tools to do all those steps, and we think there's a lack of a great open source offering in that space.

[00:45:37] JM: Again, something that's closely tied with the core Gitlab application, or is this something that's going to be like a different application?

[00:45:47] SS: Yeah, this will be our second product, and it has nothing in common with Gitlab. We try to use Gitlab CI, but the team didn't like it. They said Airflow is much better. So we're using Airflow, we're using DBT, we're using Singer. We're using all kinds of other open source

projects, but Gitlab didn't make that list. So the team decided against that. So we'll let them do their thing. They need that freedom to pick what's best for that problem, and it's a different space. It's Python, not Ruby. Then there're different tradeoffs that people have to make. We want to start simple like we started with Gitlab.

[00:46:19] JM: Okay. So for all the products you could have chosen to – I think going after a second product sound like it makes a complete sense right now. You've got an enterprise product that's got really good revenue properties. Why not have a second product? Of all things that you could have launched as the second product to go after why the – I'm still having trouble understanding exactly what it is that – What is it? The API request? Audit trail thing?

[00:46:49] SS: Yeah. Why go from the data lifecycle the way from API to dashboard? It was because I was surprised when we needed something like this. So Gitlab is growing. We got more and more data. We wanted proper dashboards for reports to follow over performance indicators.

[00:47:06] JM: I see.

[00:47:08] SS: First of all, it took me a while to just figure out what all the different steps are of a data pipeline. There's so much literature. I do still need transformations if you have more link, the answer is yes. But it's hard to understand. Then I finally had an idea of all the different steps, then we'd looked at, "Okay, what's the best open source software for each step." That took a whole while, and there was just obvious gaps. Then integrating everything together was an enormous work.

So between those three things, figuring out what the steps are, getting the software, and then integrating it, it took so much time. I was like, "Well, if other people are going to the same thing, like there's an obvious need for something that's integrated that has the best open source integrated by the fold and that's kind of opinionated about how you do this.

[00:47:55] JM: So the problem is dashboarding? Explain to me again. Sorry. I'm slow on this.

[00:48:01] SS: No problem. It took me a while too and I'm probably not doing a super good job of explaining it, but the problem is you have some kind of data source. For example, you have Salesforce and all your customer data is in there. You want to have a report of a certain type of like revenue by like when they first purchased and then by kind of region in a county. Just doing that and getting the data from Salesforce and maybe doing this for, for example, Google Analytics or something like that, it's hard. You need to clean up stuff. You need to get the data from the source in the first place. All those steps, you'd think that that'd be a few minutes, but it takes a couple of hours. The extraction and loader tools are not integrated with the transformation tools or not integrated with the modeling tools and are not integrated with dashboarding and notebooks. There's pretty great commercial solutions, but I do believe that open source is going to have a significant chunk of this market as well.

[00:49:11] JM: Interesting. It's a good point. I can see the parallels between the core Gitlab product and what you're trying to – What is it called? Moltano?

[00:49:23] SS: Meltano.

[00:49:23] JM: Melta – What does that mean?

[00:49:26] SS: Model, extract, load, transform, analyze, notebooks and orchestration.

[00:49:33] JM: Okay, cool. That makes sense. Because I've talked to a number of different companies who have a data platform, and in one sense they want to treat it like a fully integrated thing, like Uber has a data platform and you can zoon out and it's got HTFS and Presto and Spark and all the different parts of it, and it's got the OLTP database that they're taking all the data from and transforming it and then allowing other data scientists and stuff to run all kinds of cool jobs and dashboarding and stuff on it.

So they want it to be like a fully integrated entity, but they also want it to be composable and they want to allow people to pull stuff out that doesn't fit for them and replace it with something else. Sounds a lot like the core Gitlab workflow for that other part of the software lifecycle.

[00:50:33] SS: Exactly, and we always wanted to be able to not use a part of it, but I do think there's room for like a convention over configuration solution for the data space that just makes these steps easier to do.

[00:50:47] JM: Right. Are there any products that's – Because I've done a bunch of shows about the "data platform" space and it seems like a lot of people has that same sensation that they're should be an easier way, there should be a convention over configuration, and yet there's a few companies who I could say have kind of taken a stab at it, but are there any companies you admire in this space?

[00:51:14] SS: There's a lot of companies, ones that come to mind right now is Datarama, another one is Stitch Data, and I think they have great products, and Stitch, for example, they open source Singer. It's an extract and load tool, and we're using that as a part of Meltano.

[00:51:32] JM: Do you have a go-to-market strategy for Meltano yet?

[00:51:35] SS: Not yet. Right now it's still very early days or it's completely open source and the first thing we want to see is more people using it.

[00:51:43] JM: Okay. I want to get a feeling for your management strategy. First of all for people who don't know, it's a remote organization, and I guess first question, you've scaled up over the last year from how many? I know you're at 500 employees now, but how many before that? A year ago? A year and a half ago?

[00:52:05] SS: Yeah, last year we grew from like 200 to over 400.

[00:52:09] JM: Okay. Did the remote model have any issues? Did you have to do any kind of adjustments to the remote model over that course of that doubling?

[00:52:21] SS: Not really. It made it a lot easier because we already had a lot of things written down. So we're much more effective in onboarding new people, because a lot of the knowledge is already there.

[00:52:33] JM: Do you still interview every employee?

[00:52:35] SS: We don't. I stopped that [inaudible 00:52:36] 150 people with that. It's either me or Carol who reviews every last offer before it goes out the door. So we review the hiring packet.

[00:52:47] JM: How are the hiring processes changed?

[00:52:51] SS: There's a much better division of responsibilities. So we have sources, we have recruiters, we have candidate specialists, we have a person who's almost specializing now in compensation. We have a better idea of like the gearing ratios, like how many sources do we need for how many vacancies so that we have an off capacity.

[00:53:12] JM: So last time I went through a hiring process for a tech company, it was this terrible byzantine process of whiteboarding interviews and boring boiler plate questions. Every interview process I've ever gone through, in fact – Well, almost everyone has followed that recipe. Do you follow that convention as well? Do you think that's the way to do programming interviews? Is that the best way we know?

[00:53:37] SS: One of our sub-values is boring solutions. So we don't try to reinvent the wheel, but in interviewing –

[00:53:43] JM: Not this wheel.

[00:53:45] SS: In interviewing, we don't think that a boring interview is very good, and we don't believe in whiteboarding. Whiteboarding also is not great for hiring more diverse set of people. What we do do if you interview as a software engineer, you get to work on the actual problems. So we offer you to work on an actual feature in Gitlab. If you don't want to do that because you feel like you're giving us free labor, that's find. You can work on something else. We're not doing it for that reason. We actually spend a lot of time on guiding people there. If you like to, you can take up real issue and then work with our engineers to try to solve that. So it's not a whiteboarding interview. It's doing the actual work interview.

[00:54:29] JM: How do you divide up the teams within Gitlab?

[00:54:34] SS: So we have a functional organization. We have two products, Gitlab and Meltano. Meltano is a team of seven. It's kind of a cross-functional team, a typical kind of startup. The rest of Gitlab is completely functional comparing our success to a company like Apple, but for example Apple has the same thing where there's a software and a hardware division and they are responsible for their own parts.

We think that's great, because that advantage of a functional model is that you have a boss who really understands what you do day-to-day. One of our values is results, and by results we mean we measure your upward nurture input. So we don't track your hours. You don't have to ask for permission to take day off work, to take vacation, but that means we do need a manager that understands how much you're producing, because that's the way we manage the company, and that's been great. When we have something that requires a cross-functional set of people, we make a group, but there's no boss of that group. Each reports to your individual manager.

[00:55:39] JM: What's the interaction between internal engineering teams at Gitlab and the open source community?

[00:55:47] SS: So we're really proud that we work in our open issue trackers. So everything we do, except for security vulnerabilities that were reported to us, everything else is on our issue trackers. Sometimes engineers who just joined Gitlab are like, "Who's this person who is giving me all these instructions? I can't find them on our team page," and we're like, "Are you sure they're part of the team? Because sometimes they're just a very enthusiastic user who has an opinion about what the person is doing."

[00:56:17] JM: What's your approach to senior leadership structure?

[00:56:22] SS: Yeah, we have an executive team and we try to keep it simple. So if people are responsible for like sales, marketing, engineering, product, people, or finance, or alliances. It's very important to me that – It's pretty clear what each person is accountable for.

For example, product is responsible for planning what we're going to make and when we'll make it. Engineering is how we'll make it and delivering with a lot of velocity, all the feature velocity.

Marketing is responsible for generating pipeline. Sales is responsible for closing that pipeline. That leads, for example, you have a question whether sales development representatives, people that reach out to companies, whether they should be under marketing or sales, or they have to build pipelines [inaudible 00:57:13] marketing in Gitlab, which means if marketing can make the tradeoff whether to hire more SCRs or to invest more in advertising campaign.

[00:57:23] JM: Tell me about how your perspective of investors and investment capital has changed in the last couple of years.

[00:57:33] SS: I had a few preconceptions before I came to the valley. I read a lot of blogs. I'm still a big fan of like DHH and people like that, and I told my wife, "If I want to raise external capital after Y Combinator, stop me." She saw the transformation during Y Combinator. If we wanted to succeed, it's easier to not raise venture capital, because your odds of success go down from 90% to 10%. But it turns out, I was really into just doing something interesting. If you want to do something interesting and strive for it to build a really big company, in order to hire the people you need for that, those people want stock options. You cannot hire great talent and evaluate giving out stock options. Those stock options have to be worth something someday, and it's better sooner than later. So you got to grow really, really fast. To do that, you need money. So I kind of figured it out and I've been surprised how great investors are. Maybe we're lucky because we've been successful company, but almost every investor, even the ones that like declined us has been very thoughtful and kind and attentive even when we were still starting up. So the level of knowledge and professionalism really surprised me.

Yeah, for example, we have an investment from Khosla Ventures. It's ran by Vinod Khosla. The guys is like a multibillionaire. He can do whatever he wants. At one point during a negotiation, I asked him for, like, "What should we do here?" I was like, "The other venture firm wants X, Y, Z." He said, "Well, what do you want to do?" I said, "I think we should do this." He's like, "Okay. They're wrong, but I'll let you do it anyway."

So I went back to our lawyer, I'm like, "You told me this was – Venture firm X, Y, Z was right. We better call everyone to figure out what's – This guy, Vinod, is an expert." So we called everyone and it turns out Vinod was exactly right. But he like proved an unreasonable demand of

pragmatism, like he wanted to support the company, wanting to support the leaders. I think that's great. I think it's great if you can make that distinction.

So venture capital is very counterintuitive and I respect a lot how people do that. When we were raising with Gitlab in the beginning, I pitched someone at evaluation of $7 million and it went so well that for the next pitch, our second pitch, I increased evaluation $9 million. I was like, "$7 million is too little." They were so [inaudible 01:00:25]. Then they ended up passing. Luckily we closed some of the others. Then two weeks later they come back and they say, "I couldn't sleep about this. I want to invest." I was like, "That's great, but do you realize what evaluation is now." He's like, "Yeah." I said, "It's 15 million. It's more than double." He said, "Yeah, I realized. We still want to invest." At that point, he took kind of a loss to what he could have done. That money relative to the alternative, lost. I think it's psychologically really strong that you can get over that and still make the investment.

[01:01:05] JM: Has the value of Y Combinator scaled as the size of your company has scaled, or is Y Combinator – Do you feel like it's an institution that's disproportionately valuable for an early stage company?

[01:01:19] SS: They had the biggest impact on us when we started. We wouldn't be here today where we are without their help. It's great if you're technical cofounders, like we were. It's like it's ideal for that. You learn a lot about marketing, about sales. What they also added to us is the sense of ambition. We were planning to be a company of 20 people in 5 years. At Y Combinator it's like, "No. You should only stay in the valley if you want to be a billion dollar company."

Also, we started to iterate much faster, because all the other people in our group were going much faster than we were. So just kind of the healthy peer pressure to start iterating faster. I do think that Y Combinator is getting better and better at giving value later on. So they now have an A round thing. They have a grow thing where they help these companies through these stages and that's looking to be very successful. So they'll keep adding value. But for us, the biggest thing we got out of it was seeing our peers and seeing how much faster they were moving and we were like, "Wow! You can move that fast?" So we drove back home and told our coworkers, like, "We got to up our game here," and it's not that we started working way more hours. It's more that we said, "Okay, instead of a two-month plan, what can we do next week that will have

an impact?" That was something that really changed what we did, and that's where our iteration value came from.

[01:02:45] JM: Okay, last question. I have been intrigued for a while about the Github for X type space. So Github for artists, Github for musicians. I don't know if you've seen the company Splice, which is a Github for musicians attempt. Most of these don't seem to work. There seems to be something, either it's the technology of git or it's the characteristic of the software engineer role that makes large scale decentralized collaboration work particularly well for software engineers.

Do you think it's like – Doesn't it surprise you that there's not more collaboration on the internet over artists, musicians, people who are not programmers? Why do you think that is?

[01:03:38] SS: I think a huge part of that is a technical reason. You can only do distributed version control if you can merge things back together. If something is a binary, like a Photoshop file, or like a sound file, you can never merge those back together and you end up with garbage. So you need like a human readable file of like the information, and that's not available for a lot of things. If you added a movie right now, that editing information, if you can even get it separate, it will just be binary. You can never merge it back or make a suggestion to it. So that's a technical reason.

I think the other reason is that a lot of these things are not truly distributed. They're a SaaS service where you all have to be on that platform, and I think a big thing of git is you can just pick up your bags, and if you joined Gitlab, you can just run to Github import and we can import everything, all your git repos and we can even do issues and milestones and everything else. I think that makes people a lot more comfortable with adapting something like it. I do think there's some awesome collaboration tools. A company that's also all remote and even bigger than us is Envison, and they seem to be very successful with their collaboration software for designers.

[01:04:55] JM: Absolutely. Sid, thanks for coming on the show. It's been great talking.

[01:04:57] SS: Likewise. Thanks for having me.

[END OF INTERVIEW]

[01:05:03] JM: This podcast is brought to you by wix.com. Build your website quickly with Wix. Wix code unites design features with advanced code capabilities, so you can build data-driven websites and professional web apps very quickly. You can store and manage unlimited data, you can create hundreds of dynamic pages, you can add repeating layouts, make custom forms, call external APIs and take full control of your sites functionality using Wix Code APIs and your own JavaScript. You don't need HTML or CSS.

With Wix codes, built-in database and IDE, you've got one click deployment that instantly updates all the content on your site and everything is SEO friendly. What about security and hosting and maintenance? Wix has you covered, so you can spend more time focusing on yourself and your clients.

If you're not a developer, it's not a problem. There's plenty that you can do without writing a lot of code, although of course if you are a developer, then you can do much more. You can explore all the resources on the Wix Code's site to learn more about web development wherever you are in your developer career. You can discover video tutorials, articles, code snippets, API references and a lively forum where you can get advanced tips from Wix Code experts.

Check it out for yourself at wicks.com/sed. That's wix.com/sed. You can get 10% off your premium plan while developing a website quickly for the web. To get that 10% off the premium plan and support Software Engineering Daily, go to wix.com/sed and see what you can do with Wix Code today.

[END]