EPISODE 1300

[INTRODUCTION]

[00:00:00] JM: If you've ever googled a computer science or programming question, you likely found an answer, or multiple answers on Stack Overflow. It was founded in 2008 and named after a common computing error, the Stack Overflow. Stack Overflow provides technical recruiting, advertising, market research and enterprise knowledge sharing to its over 15 million monthly users. Stack Overflow for Teams is a knowledge management and collaboration solution that developers and managers already know and trust.

With Stack Overflow for your specific company, you can onboard faster, drive employee engagement and improve productivity by cutting out the chat threads. In this episode, we talk with Tom Limoncelli, a Manager at Stack Overflow, an author and a tech advocate. Tom is a wonderful guest. He's a highly experienced engineer. A lot of this episode is talking about the utility of adopting new tools, and using those tools to transform processes within your company.

It's a very good show on knowledge sharing. Also, full disclosure, it's sponsored by Stack Overflow Teams. It did convince me of the worth of Stack Overflow Teams. I'm planning to try it out in my new company, Super Compute. We're going to figure out how to use Stack Overflow Teams internally. It seems very useful. Or, for example, building a very complex game engine. Internally, the game engine is a ball of spaghetti code, and that's the way it has to be. It's not exactly spaghetti code, but it's hard to work with. The idea of having a dedicated Q&A platform is quite useful to me.

In fact, I'm going to go and create it right now. I hope you enjoy today's episode.

[INTERVIEW]

[00:01:36] JM: Tom, welcome to the show.

[00:01:38] TL: Thank you, Jeff. It's a pleasure to be here.

[00:01:41] JM: You work on a Q&A product called Stack Overflow for Teams. My question is around the adoption of internal Q&A software. There are de facto SaaS tools that have become part of every standard engineering stack. There's Slack, there's GitHub, there are a few other – I think, Sourcegraph is becoming a standard. Why is the domain of internal team-based, company-based Q&A a product that should be in the canonical SaaS product suite?

[00:02:26] TL: Well, Jeff. I think, the answer is that software development is a team sport. As part of a team, you need to communicate and you need to be able to record information that lasts longer than any one person stays at a company. Slack is good for real-time communication, but when you – the benefit of Q&A system is it's permanent. It's **[inaudible 00:02:54]**, and lots of other reasons up to.

[00:02:58] JM: Why is any different than having wiki software, or advanced wiki software, like Notion? Can't I just store all the necessary FAQs and Q&A and stuff in there?

[00:03:10] TL: Good point. Excellent question. I know, I've been a wiki fan since 2000-2001. I was an early-ish adopter. I love wikis. One thing that I've noticed about wikis is often, they stall out. People stop entering information in them, or someone writes a really good wiki page, but no one updates it. I don't know. I've talked to people. Some people are afraid to change someone else's document, or they feel like, “Well, it's not my place to change another document, a wiki page that someone else wrote.”

In the Q&A format, it's always polite to ask another question, or put in a comment. We have an articles function, which is basically a wiki. We call that the community wiki, because we want to encourage people to edit that document and keep it alive.

[00:04:10] JM: I know how normal Stack Overflow fits into my workflow. That is, if I have a question I don't know the answer to, I find the answer to it on Stack Overflow. I do not ever commit any knowledge to the world of Stack Overflow. I think, there's a lot of people like me. Part of the reason for that is that I'm a little bit – I have stage fright. I don't want to go out in front of the entire world and say, “Look, I know the universal –” Because that's what Stack Overflow is. Stack Overflow is, this is, you are basically going up against the Q&A viability of every other engineer on the Internet.

If you're doing it within the domain of a company, it's a lot more subjective. You're going to be more of a domain expert. It's more of an internal collaborative safe zone. It seems like it's a very different experience than engaging in Q&A that is Internet-wide.

[00:05:11] TL: Yeah. It's so very different. Yet, it's so very much the same. I'd like to think of it as if you're a Python programmer and you have a Python question, sure, you can ask that question on Stack Overflow, and you're going to get an answer. If you're a developer at a company, and there are companies with 20, 30, a 100,000 developers, which that's larger than the Stack Overflow user base 10 years ago. Those developers have questions that are possibly about some internal proprietary system.

Maybe you're the maintainer, or maybe you need to use some internal authentication library. Obviously, you can't post that on public Stack Overflow. Those people don't even have access to that library. You can post that on Stack Overflow for Teams, because it's private. One way that it's different is just the – you have a secure place to ask questions that can't be asked in public. Then the other way it's different is, yeah, the audience is different. It's your co-workers and your co-workers, I find, are more likely to be more collaborative, more friendly, so to speak. It's actually easier to ask questions there.

You mentioned that you're doing this in front of the whole world with Stack Overflow. Oh, no. You use word, the phrase, 'stage fright'. Yeah, I feel the same way. On public Stack Overflow, I'm cautious about asking a question. On Stack Overflow for Teams, I see very different kind of questions asked. Yeah. Sure, there's the technical questions. One of my favorite questions on stack was someone asked, "Hey, we all use this particular tool. Does anyone have any tips about how to use it better?" There were dozens of really interesting write-ups. I learned so much from reading that. Some of them were technical. Some of them were just surprises like, "Hey, if you press Shift while clicking that one particular button, the feature acts totally differently, and here's the cool things you could do with that."

That's not the question that people frequently ask on Stack Overflow, or actually, Stack Overflow Public is actually, they discouraged questions that are opinions. It's supposed to be technical, canonical questions and answers. You have more flexibility when it's Stack Overflow for Teams, when it's your own personal Q&A system.

[00:07:45] JM: Slack has been a work in progress for the software product development lifecycle. We have learned over time the norms of Slack, and Slack UX has adjusted itself to be more in-line with the norms. What are the norms that develop around a Q&A product?

[00:08:09] TL: I'm not on the design side. As a user, I'd say, one of the norms inside stack is when someone asks a really good question on Slack, often, the response is, “Hey, that's a great question.” Or actually, when someone writes up a great answer on Slack, the response is often, “Hey, that's a great answer. You should really record that in – let's memorialize that by putting that up on Stack Overflow for Teams.”

Actually, we have an integration with Slack, so that makes that easier. In fact, it basically works like someone will ask a question in Slack, and then someone else will use the integration and basically, click a button that the bot jumps in and says, “Hey, I see you're asking a question. Would you like help transferring that over into Stack Overflow for Teams?” That actually helps prime the pump and move the conversation to Stack Overflow for Teams. As a result, it's not just one person asking the question in a particular Slack channel, but it becomes more searchable. Other people might jump in with answers. That's one norm.

Another norm is just the fact that there's different – In my mind, there's different kinds of questions. There's the specific technical issue question. There's the general like, how can I use this tool better in question? There's also more open-ended questions, like, what's the history of this? Why was this designed this way? That's a question that you might not see on Stack Overflow public. At stack, frequently, someone will say like, “I know that we do this a certain way. How did that happen?”

People that have been at the company for long time can chime in. People have written long essays explaining the history of things, which I always laugh, because as a manager, if I had assigned to that person, you’re like, “Please, I would like you to read an essay on why this is, and put it in our document repo.” No one likes to write documentation, I find, especially when your boss tells you to do it. When a co-worker asks, “Hey, what's the history of this?” People will just write and write and write and you learn so much valuable stuff that way.

[00:10:36] JM: Yeah. There are different cultures, different companies have different cultures around the written word. Amazon has this culture around the memo, the six-page memo culture. I think, that works great for Amazon. Maybe it would work great for every company, but it doesn't seem like every company wants to do that. The format of Q&A is a compelling – I mean, as a podcaster, is obviously a compelling domain, because podcasting is entirely Q&A. I know, the value of Q&A is it's the socratic dialogue. There's a there's a reason the socratic dialogue is a big deal. It's you have a question-driven, curiosity-driven narrative structure, or way of architecting the information systems in your company. It's a very natural form.

That to me is a desirable way. It's a desirable means of communication in a company. That's not to say, you don't want Readme.io, or you don't want Confluence, or whatever. Having a dedicated place that encourages and facilitates Q&A seems pretty useful.

[00:11:46] TL: Sure. One nice aspect of that is it's a – Q&A sites acknowledge that there's more than one answer. The word I like to use is it democratizes education. If I think back to my education, so much of it in school was a lecturer standing in the front of the room, telling 20 or a 100 people, this is the way things are. Questions were supposed to be to clarify what the person said. That is a very different mode than a Q&A website, where education is democratized.

First of all, there's more than one right answer, right? Someone might say, “Well, I wrote that library, and this is what you should do.” Someone else might post another answer that says, “Yeah, I agree. But this is the code snippet that we use every time, and it's better for these reasons.” Now you have a dialog. Also, the example I gave before with what tips do you have about XYZ? You’ll get dozens of answers.

What I really like about that is the fact that it breaks this tradition that some companies have that senior people teach everyone else. Instead, replaces that with a model where everyone teaches everyone. It really democratizes the education within your company. This is a very long answer to a very short question that you asked, but I also think back very early in my career, there was a time where I went to a training class. It was a three-day training class. I learned one thing. I mean, I went there, because there was one particular question that I wanted answered, and I did get that answer and it was great.

Think about it. I had to wait a month for the class. The class was three days. The one thing that I really, really, really want to learn took about 15 minutes out of those three days. Compare that to how we want to learn today. Learning is on-demand. It's instant. I will write a question in Slack, someone will hit that button that says, “Hey, this would be – I don't determine that this would be a good thing to list in the Q&A section.” A different coworker says, “Oh, yeah. That's a great question. That's not just a great question. I want the answer memorialized.” They click that button. It goes into Stack Overflow. Now we're writing it there. I'm not just getting one answer; I'm getting multiple viewpoints.

I can upvote the one that was, or multiple items that were useful to me. I can also tag it. I can make it more searchable. That's just the question asked, or point of view – from the answer point of view. I was speaking to an engineer a couple months ago. He maintains a library inside a very large bank. For years, he just maintained this library, or him and his team maintained this library. It's a lonely thing, right?

They knew people used it, only because people complained about it, they would file bugs. Once it got Stack Overflow for Teams, they registered a tag related to this library that they maintained. Part of his morning ritual is he comes in in the morning and looks for new questions that are tagged with that tag. Now, the questions are flowing to him. He feels so much better about his job, because he's helping people. He begins every day helping people use his library better, and get feedback about how it's used and where it's used and how he can do a better job. This guy says, it's actually helped his morale. I'm just fascinated with that story, because it's so much the opposite of the lecturer standing in front of class, telling people how things ought to be.

[00:15:45] JM: Let's talk about engineering a little bit. Presumably, there is some code that you can take from core Stack Overflow, and put into Stack Overflow for Teams. What's the migration story there? How much code were you able to borrow? What did you have to write from scratch?

[00:16:05] TL: Oh, excellent question. The engineering story is very interesting. First of all, our goal is to have one software base, one big git repo that – well, it's actually multiple small repos. One collection of repos that you compile it one way, and it's the public Stack Overflow. You compile it another way, and it's our enterprise product, etc., etc. We try to keep the – for the programmers out there, the #ifdevs for lack of a better term, as small as possible. It is one big codebase that does many different things.

That's been some challenges, because we actually have for the different ways that we build the product, some of them have slightly different database schemas, or certainly, authentication is different, etc. There's a, I wouldn't say a plugin architecture, but definitely – well, with a little hand waving, I'll call it a plugin architecture. One of the nice things is, since we started with a codebase that scales to millions of users and billions of questions – Wait. I think I reversed that. Millions of questions and billions of users. Stack Overflow, the business product and other things, scale is less of a problem, because we're starting with a codebase that is for a huge number of users and we were scaling it down.

[00:17:34] JM: In terms of company strategy and resource allocation, you've got this core product of Stack Overflow. There's plenty of engineering work to be done around that. How do you divert engineering resources from that core product to a new product?

[00:17:55] TL: Oh, that's an interesting question. My boss's boss is actually the expert in that. She's really changed the org structure to be better for what we're trying to do today. I won't go into the details. The first thing is, the way you phrased the question, Jeff, sounded like, are we stealing developers off for one thing in that starving products in other areas? I don't think that was your intention. I just want to be clear that we're hiring like crazy. People are shifting around.

If I had to summarize it in a nutshell, I'd say, we're organizing, so that there's a team for each persona, so to speak. There's a team, instead of a team that's either trying to do everything, like instead of a huge team that tries to do everything, or a couple small teams each responsible for a different technology. Instead, the teams are organized around users. For example, there's a team that's focused on the onboarding process or sub team. There's a sub team that's focused on billing, for example, or the Q&A experience itself, the actual act of asking a question and answering a question.

That way, each team becomes focused on instead of focusing on the technologies, they're focused on the users. They're learning what the users need, how they act, how they interact and becoming experts in that, then writing the code is really a sub-problem, if you think about it. Once you understand your users, writing the code is the sub-problem.

[00:19:38] JM: You worked at Google for eight years was it?

[00:19:45] TL: Seven years.

[00:19:45] JM: Google for seven years and then Stack Overflow for eight years, right?

[00:19:49] TL: Yeah. Bell Labs for seven years before Google.

[00:19:52] JM: Epic. How has your perspective on management changed in those different zones, in those different time periods?

[00:20:06] TL: I'd say, all three had different variations of hire smart people and get out of their way, but at different scales. I would say at Google, managers spent a lot of their time dealing with the politics, the internal politics to shield the engineers, so they didn't have to deal with them, or in many cases, were oblivious to them.

At Stack, the focus has been more on getting to know the engineers, getting to know how they work and what the roadblocks are, and focus your time on clearing the roadblocks. Sometimes helping them to clear roadblocks. Sometimes you're teaching someone to fish, sometimes you're doing the fishing. Stack is very much a company that believes in servant manage. Now I'm blanking on the term.

[00:21:03] JM: Servant leadership?

[00:21:05] TL: Servant leadership. Thank you. Yeah. Which is funny, because I've talked to a lot of people, a lot of companies that use that phrase, but I've never seen it emphasized so much, than at Stack. I mean, I really feel like the managers at Stack, they're interviewed for that management style. We have the internal tools and support to be able to really do that. As an example, if two engineers came to me saying, “Hey, we've been arguing for a week over. Should we do it this way, or that way? Why don't you make the decision for us?” My response would be, “I'm not going to make that decision for you. I'm going to help you with the communication. Let's sit down. Let's work out the pros and cons. I'm going to help you make that decision. Because you're the engineer, you should be making that decision.”

I mentioned, if managers made all engineering decisions, well, that would be a disaster. I might have great ideas. I mean, I've been an engineer. I am an engineer. I still write code, in fact. But the real engineering decisions have to come from the engineers.

[00:22:14] JM: The history of Stack Overflow is that it came out of Fog Creek Software, right? That's the name of the company. Fog Creek Software birthed Stack Overflow, Trello, Discourse, Glitch. Four of them? Was there four of them?

[00:22:33] TL: Yeah. I don't know if Discord –

[00:22:37] JM: No, Discourse.

[00:22:38] TL: I don't know if Discourse counts. Because at a certain point, we were separate enough from Fog Creek that I can't accurately recount the history. Yeah. Fog Creek did spawn many different projects, Jeff.

[00:26:38] JM: What was it like in the, I guess, the ivory tower – Not the ivory tower. The Googleplex. The insulated Googleplex, the firewalled Googleplex, firewall of ideas.

[00:26:49] TL: That’s an interesting story, because, yeah. I was in the Google bubble. We were certainly having amazing conversations internally and learning a lot of stuff. All the stuff that we now call DevOps was being discussed using different terminology. That's why the S3 versus DevOps divide happened, because in my opinion, Google was insular. Actually, when I was no longer at Google I've written some books on IT and IT topics. Well, after Google, I said, I really need to write a book that's like the cloud computing book.

Google had just started being more public about their internal techniques and policies. I was afraid that if I use the phrase SRE in my book, I might get in trouble. I don't know. In hindsight, that's silly, because what do you mean? Someone's going to sue you over an acronym? Then Google's started to be much more public, this, they helped create SREcon with the USENIX Association. They started publishing these books and everything.

The world of Google terminology had to meld with the independently developed DevOps terminology. It's like how two different people invented calculus at the same time. There was just a desperate need to do that math and what's the phrase? Necessity is the mother of invention. Similarly, DevOps and SRE came about in parallel. Now, the terminology is merged. Yeah, it was really weird.

When I left Google, I spent a month in the wilderness, asking people all these questions that must have sound crazy, because I was like, “What does the rest of the world call this thing?” Yeah, it was quite an experience.

[00:28:46] JM: In 2014, you published practice of cloud system administration, the DevOps and SRE practices for web services, Volume Two. Or actually, was at the first one in 2014?

[00:28:57] TL: The titles, yeah, naming is hard. Actually, this is the 20th anniversary of my system administration book. In 2014, we called the next version – we wrote from scratch a new book, which is the one that you just mentioned. We called it Volume Two, because it's the sequel to the sysmin book. Yeah, I'm sorry. I think, I lost your question.

[00:29:20] JM: Well, first of all, congratulations. I am publishing my first book, actually, on July 6th. It was an arduous process. Congratulations on writing a book, multiple books.

[00:29:31] TL: Thank you. You too.

[00:29:33] JM: Thank you. You're writing about cloud system administration. If you've been thinking about sysadmin for 20 years, you have seen history repeat itself more times than I have. I've been in the industry basically, for a decade. Today when I build software, I am looking for the highest level available building blocks. If I can get a SageMaker-managed machine learning system, I'm going to use that. If I can get a completely managed database, I'm going to use that. I don't want to be thinking about operations. I don't want to be thinking about inserting agents, if I can. I want to do a one-click, high-level, like yeah, put the agent on my container sort of thing. Is it notably new? When you think about the timeline of development, the level of abstraction that we're working with, where so much of the lower level plumbing has been pulled away, does it feel definitively new to you? Or does it feel just like a natural next step?

[00:30:34] TL: Jeff, it feels new every day to me, because – and rather than looking at it as we keep reinventing the wheel, I look at it as this long, slow, or punctuated series of changes that are all in one direction. That direction is higher levels of abstraction. As a second place, I'd say that the other big change is scale. Scale keeps getting bigger and bigger. My first job out of college, I thought my boss was the best system administrator ever.

One example I would tell people is back then, you didn't buy an ethernet cable. You bought the cable and the connectors and a little device that would crimp the end connector onto those what, six or eight wires that are in the cable. He could crimp them and get it working on the first try. I would crimp and it wouldn't pass the test. Yes, you had a little machine that would test your ethernet cable. I would have to crimp two or three times before I got it right. I was so impressed that he could get it right on the first time every time.

Fast forward 30 years, who even thinks about making their own cable, right? You order at 3-foot, 6-foot, whatever length you want. There's a variety of lengths. Actually, silly me, who even orders cables anymore, right? If you need to cable up a machine, there's an API call that makes a machine in the cloud. You don't even care how it connects to the network. A lot of system administrators don't even learn the fundamental, or low-level networking that I had to learn, because computers just magically talk to each other now. You're concerned with what's the latency, not how it gets there.

Yeah, we just keep moving further and further away from the hardware. Thank God, because who wants to sit all day racking, stacking machines? I would rather have an API call that spins up a machine, have my little Terraform script that configures the network and sets up machines in a CICD system that deploys software. I mean, when I got started, deploying software was often like it would – the first step in deploying software was picking which weekend you were going to kill, because you were going to spend that weekend deploying, doing a big release, and hoping that if we get it all done on Saturday, we don't have to lose our Sunday also. I'm so happy that we don't have that anymore, or I mean, that I don't have that. I know, there are still companies that are in that situation. Yeah, I think this is all progress.

[00:33:18] JM: I've been in a data center only a few times in my life. One of them was a few weeks ago. I was walking around this data center with a friend of mine. He was telling me about what a company like Google has to do to avoid, for example, noisy neighbor problems, and all of the cascading failures that can occur from noisy neighbor problems. Where if you misplaced some workloads, you're going to have them too close to one another. They're going to both be noisy. They're going to cause a problem. Then if your scheduler is operating in some peculiar fashion, you might have the scheduler reschedule those workloads and create more noisy neighbor problems and just basically corrupt everything. You have some crazy downstream situation and take down a data center.

I think that's an extreme example, but it illustrates the level of sophistication that operating these data centers can entail. Do you have any insight about – I know you haven't been at Google for eight years or whatever, but just as a theoretician who's written multiple books on this topic. I know this is not exactly the domain of sysadmin, but seems it overlaps, or has some overlap. I'm sure you've spent some time talking to data center people. Can you tell me about but how hostile is that environment these days? Or has it been like, have we operationalized everything, abstract away all those problems?

[00:34:45] TL: Yeah. Well, first, I should say, my last time in data center was yesterday, actually. Stack Overflow's, the public website is still in the data center. That could change in the next couple of years, but we still do some bare metal work. As far as problems, like noisy neighbor problems at Google, so when I was leaving, they were doing stuff where containers were avoiding the noisy neighbor problem by – this has all been merged into the Linux kernel now, so it's public. Containers can actually lock certain rampages, but also network bandwidth and disk IO and stuff.

That's avoiding the noisy neighbor problem at a very low level. Recently, Google's published some papers about avoiding at the high level. Actually, I haven't read the papers yet. I read the first half of some of the papers, so I shouldn't try to quote too much.

[00:35:47] JM: I cite papers from the abstract all the time.

[00:35:49] TL: Okay. I think, what Google's dealing with right now is it's like, the typical person is dealing with noisy neighbors at a very high granularity. What I see as an ex-Googler, when I read their papers, what I'm seeing them doing is dealing with – they're dealing with at such a small granularity that they actually refer to it as tail latency. If you have a 100 machines doing the same thing, you're going to find that most of them are going to complete the same task in the same amount of time. Some are going to take longer. There's a great paper called The Tail at Scale, that shows that Google is thinking about this problem way deeper than anyone else. I think, it's a Google-sized problem, and most people don't need to deal with it at that scale.

[00:36:44] JM: While we're in the subject of low-probability events, I was having a conversation with somebody the other day about whether or not there's an irony here. The Internet was basically developed to be resistant to some catastrophe, like a nuclear bomb, right? I worry now that our infrastructure is so dependent on the data centers, that we actually have the inverse problem, that if one of our data centers went off the face of the earth, for one of these public clouds, that it could bring down the Internet. How much faith do you have in the security? I know that's not totally in your domain, but given that you've worked at Google and you have such a background in sysadmin, I'd love to hear your – or if you can give a take. If you don't want to, I understand.

[00:37:33] TL: Well, I think the important part of that conversation is that it's becoming more and more important to think about that thing. It's important that everyone think about that thing, the developers and operations. Thinking about that thing used to be something that we would leave to operators. They'll figure it out. You can't write code one way and expect operations to run it a different way.

For example, when we were launching Stack Overflow for Teams, we knew it was going to scale up huge, because it's now completely free for the first 50 users and we were thinking, "Oh, my

God. We're going to have huge scaling problems." In the old days, I would have sat the SRE team down and said, "Okay, what are we going to do to make this scale?" You can only make so much progress if you're working inside a silo. Instead, what we did is we sat down with the architecture people and the application people and the SRE people, and we had a bigger conversation. What are we going to change in the code? I'm sorry, at the high-level, first, how are we going to architect this? Then, what are we going to change in the code? What are we going to change in our operational practices? What are we going to change an operational code to make all of this work together?

You can only solve certain problems – I'm sorry. When you stay inside your silo, you can only fix the problems that are right in front of you. In fact, it's likely that you'll make operational optimizations that are good for you, but are a pain in the butt for the other silos. You have to work together. Now your question is, how do you deal with all these dependencies around the Internet? Our website depends on a CDN, depends on database stuff, and caches, and all these different things. Some of which are outside of our control.

Our engineers have to be cognizant of what those things are. For example, every new engineer at Stack gets a lecture, or utensil lecture from the SRE team, about what we want every developer to know about coding in our environment. One of the things is that your code is going to depend on lots of different external things, database servers, cache servers, CDNs, etc., the Internet itself. You have to write your code for three different situations. You have to write your code, assuming everything's fine, you have to write your code assuming that the database is having a problem and had to go into read-only mode. You have to write your code that is, in a way that works if the database is just down.

You can't just show an error page saying, database down, 404, or an error message, five something. That's a challenge. That's something that the developers have to consider at the very beginning of the design of a feature. Not tack it on at the end.

[00:40:35] JM: Okay, let's close off by revisiting the topic we explored at the top of the conversation, which is what you have been working on, which is the team's product for Stack Overflow, the concept that you as a, or the general you, the average software company would derive value from having a domain specific, company specific platform. Just rephrase, or reframe the thrust of why that is something that the average company, or the average team should want?

[00:41:16] TL: Sure. I'm going to answer that two different ways. From the developer point of view, we as developers, we're in a lot of pain. We spend hours of our day asking around, not knowing the answer to some question, whether it's, why is something working this way? Or how do I do this? Or how do I use this system that another co-worker has developed? That's a painful process to guess your way through it.

Back when we had two or three devs, you can shout a question over the cubicle wall and someone's going to answer it. Now, companies don't have two or three devs. They have teams of devs. Some have thousands of devs. It solves that important information gap problem. I can also answer that from the management perspective. It's hard to hire developers. There's a shortage of developers. The ones that I have, I want them to be as productive as possible.

Every minute that is spent delayed, because of a knowledge gap is a minute that I want to try to get rid of. It's not just a developer, like removing the pain from development, it's also just a productivity thing. Also, it's really fun. People talk about getting into a Wikipedia hole, where they spend an evening. You started looking for one thing and then an hour later, you're reading about, and I woke up to this, but maybe I just haven't be reading the reading the page about the all the different actors in The Wizard of Oz.

Anyway, when you do that at work with Stack Overflow for teams, you can do that same thing. It's fun to learn about what other people are doing in the company and how things work. It's not just fun. It makes me a better employee. Makes me more productive.

[00:43:14] JM: Wonderful. What a great conversation. Thank you so much for coming on the show.

[00:43:18] TL: Thank you, Jeff. It was a real pleasure being here.

[END]