EPISODE 1211

[INTRODUCTION]

[00:00:00] JM: Yelp is a crowd-sourced review platform focused on restaurants and local businesses. Originally created as an email-based recommendation service, Yelp relaunched in its modern form in 2005. At the time its focus on user created reviews and social interactions was fairly novel and made it stand out from competitors such as Angie's List and City Search. Since then Yelp has become a national brand, a worldwide brand. And as of 2021 it has over 171 million reviews on its site. The mid to late 2000s represented a time of explosive growth and profound change in the web application space. Industry leaders like Yelp had to adapt their technology stacks for the unprecedented scaling they were experiencing. At the same time the rise of smartphones led Yelp and many others into the mobile application space.

Michael Stoppelman was an engineer at Yelp during this turbulent time. He left Yelp in 2015 and now works as an angel investor. He joins the show today to talk about the engineering challenges that Yelp faced during this time and the profound changes that the industry as a whole went through as well as the history of Yelp and how they can contextualize the startup landscape today.

[INTERVIEW]

[00:01:11] JM: Michael, welcome to the show.

[00:01:13] MS: Hey, thank you so much for having me on. Really appreciate it.

[00:01:17] JM: You joined Yelp in 2007. This was early on in the life of the company. Tell me about the major technical challenges that the company was dealing with at that time.

[00:01:28] MS: When I joined, the first day of work I was told that I was going to be in charge of the search system and it was implemented with the Lucene Library. There were a

combination of search services. Some were using Solar. Some were using lucene. And every time they would get re-indexed, the re-indexing process would kick off where it would suck in information. The cron job would suck in information from the primary database to index the new businesses that had been added or new reviews that had been added. The search service would die. The way the updating worked, it actually would slow down the indexing process. And when the indexes would be switching out and the caching would have to warm up, it would actually fall out of the load balancer and there was like some sort of nagios ping that would be checking, or the Linux load balancer would send out some sort of check and it was checking every second and then it would see that it's not responding so it'd pull out a load balancer. So users wouldn't be able to get search for a period of time. So my first project was actually like dealing with that. And the initial load balancer for Yelp was a Linux box that had a dangling wire that was easy to jostle. And so we always would have to be very careful if we were working on the machines in the data center because the machine's actual cable was very loose. And so there have been moments where someone had like unjostled and taken down the load balancer. That was kind of my first week was seeing, “Okay, we have six engineers. I had come from Google so where there were like 5,000 engineers and I had been insulated from all this stuff.” And then I come to Yelp and it's like we have a data center cage and we've got a load balancer that's not fully redundant that's got a loose wire and the Lucene search indices that I was responsible for were being taken out of the load balancer whenever the search would be re-indexed. So that was my first week.

[00:03:35] JM: When did you transition to working on the more general infrastructure across the company?

[00:03:41] MS: Yeah. I mean as the system evolved, at the beginning of Yelp – Yelp was started in 2004. So it was all get your own data center space. And so we had our own cage space. We had to order our own machines. This was pre-AWS. AWS started coming into view in like 2008 with S3 or 2009 with S3 and EC2. We started to toe into that. And we actually started with a backup data center and a service called SoftLayer, which was bought by IBM. And SoftLayer, kind of the idea around it was that it was kind of like AWS but you got your own machines and you got to like be able to control it yourself, but that reality kind of soured as we

tried to use it more and more. It just wasn't working out the way that it was sold. And networking was pretty flaky in SoftLayer. And so we ended up in 2012-13, we started to do the deep dive into kind of pulling the rip cord to AWS as I saw that we were never going to get to that scale, or I would wouldn't say never, but it was unlikely that we're going to get to the scale of racking 50,000 machines a year or something like that. And so I was like, “We're not going to be able to hire the talent to do data center work at scale that a Google or an Apple would. So we should probably get out of this business.” And at the time our ops team was about seven people and I was just like our core work ethos, like we shouldn't be getting up every day worried about whether this – At the time whether the SCSI controller had a battery backup. One time we got shipped a bunch of SCSI controllers and they were like missing batteries. And so someone had to go in and replace all the SCSI adapter batteries. And then there were always driver issues and incompatibilities with Linux and the list just kept going on and on. So we’re just not at the scale where we really had a team that was like making that a well-oiled machine. And so that was kind of the data center story.

To make it clear like, I haven't been at Yelp for over three years. I left in early 2017. And so this is all Michael Stoppelman as a non-employee. Like I’m not employed by Yelp anymore. My brother is the CEO and co-founder to make it clear, and he's still a CEO. So I get updates occasionally from him, but I have not been in the business for three years. Just want to call that out.

[00:06:47] JM: And as of the time that you left, what was your strategy around AWS? Did you have a number of blessed services that you used from AWS? Or were engineers allowed to choose whatever they wanted to use?

[00:07:01] MS: So at the beginning the, AWS transition started as like the primary stuff that – The easiest wedge that I saw to get the team to start thinking about cloud was the MapReduce jobs that we were running. So pre-Hadoop people were using MapReduce as the primary kind of like data processing language for any sort of machine learning or a large-scale big data processing. And so we've internally built a tool called mrjob. It's still a popular Python library. And then we got that working on AWS and we got all of our logs transitioned to S3.

And so once the logs for the site were in S3 and more and more teams saw that you could process that data in an efficient way in AWS, more and more of our data processing jobs, which I would say accounted for 50% or 60% of the computation at Yelp at the time for search services, ads. All that was happening in the background either nightly or intraday. So that was the biggest wedge. And then it became, "Oh, we want to be able to run unit tests and regression tests and things like that." And we started to get more developer environments set up in AWS. And then really like we were kind of at the cusp of like where we were deciding whether to use Vagrant, which was a precursor to Docker. And then I was lucky enough to run into Solomon Hykes at some sort of meet-up and he was telling me about dotCloud and how this project internally, this project on Github Docker, which was part of dotCloud was taking off and he had like 150 pull requests that he was processing through. And so then I went out the next – He was in the midst of like this massive explosion of Docker, and that was three weeks from release of Docker and it was exactly kind of what our team needed to be able to get these containers going for all these different for all these different services. And so we were just in the mode of like how do we get – Do we do AMIs? And we have these big blobs that are going to be moving around the network or is there something better and then this was the something better? And I was pretty ecstatic about that and then the team kind of ran with that.

Our first kind of like foray into production, the challenge of Yelp was that we were kind of like a year or two years ahead of where the cloud infrastructure was and what our needs were. And so we ended up having to build a lot of technologies that were a little bit earlier. And sometimes we'd be in parallel with companies that were building technology that we'd be able to use later. So an example of that was Kubernetes didn't exist. Mesosphere was kind of in its initial stages. And so we actually had to build our own platform as a service. So kind of our own Kubernetes, our own Mesosphere, and we had to build all – Now you would use the Lyft project that does the – I'm forgetting the name of the project for load balancing, like an HAProxy replacement, but we had to have all our proxying layers and then quadratic back offs and circuit breakers and all of these pieces and bits that you now can get out of the box from a lot of either paid or open source platforms. We had to build all of that stuff internally with a pretty small team.

And so another thing that was also like we were a little early on was Kafka had been open sourced. Confluence, the company that now has taken the reins over over Kafka. But Kafka was just coming out. There were no data connectors written so we were always building every data connector and we actually open sourced the system that did all of the log stream, all of the log streaming out of MySQL to take the replication log out and then be able to put all of that into Kafka topics. All of that stuff we had to build ourselves.

And then even when we used commercial products like Splunk, the data connectors for MySQL and Postgres and Kafko were kind of like not written or written poorly and we had to write our own. So all these – Every time you took on another software project it meant another team of four engineers working on something, something resilient. And so I guess there're lots of places to dive in there and I'm happy to take the conversation wherever your head's at.

[00:12:19] JM: You really have illustrated how the pros and cons of being early to market there. Obviously the pros are that Yelp was early to market and had a chance to compete really in this really, really valuable space, but the cons were that this was an internet that seems so primitive compared to what we have today in terms of infrastructure having to build all this stuff around containerization and Kafka. And not only that, once you build it all yourself, you have Kubernetes come out, and I assume the Kubernetes platform was effectively better than what you had built at Yelp and then you're faced with the decision of do we tear out all the technology that we've built in-house and replace it or do we just kind of run with what we've built in-house?

[00:13:14] MS: Yeah. There are a number of other questions going through engineering leaders' heads when you kind of break that down or folks that are picking – Any architects that are picking these types of software products is do you want Amazon to have total leverage in every negotiation that you enter in for the cost of your cloud if they know that you're on every bit of AWS and there's no way for you to get out? They have a lot more leverage in negotiations. And so that's always running through your head.

And also like every bit of technology, the internal organization rejection of when a new product comes out on AWS, let's say, like managed Kafka and you have an internal team that has an identity with being the Kafka team internally at a company, it's not an easy decision whether – As an engineering leader whether you move on from that internal project and move the engineers to something else, or there's always internal pushback whenever a new product comes out. And a lot of times that product isn't Good enough yet but will be in a year. And so you're thinking in your head, “Hmm, like it will probably get better over time, but I don't know.”

Sometimes, like a lot of examples, there will be gotchas and the details of some of these launches of I think that the managed service of Kafka had like message limits or it had different limits on – When it released it had different limits on message sizes or kind of like replication, replication factors that we had in our own internal versions. And so there are always like limits or changes that you have to go into, but then you don't have to do the SRE DevOps work of managing these things if you use the managed service.

So I think the big question for anyone operating in these types of conditions is like if you build something that doesn't exist in a cloud and then inevitably over the next three or four years something pops up that does that thing and you have a team around it, are you willing to really re-examine yourself, your organization or your own sub-positions about what you've built and kind of replace it if it's cheaper or easier to manage? I think that if you look at any enterprise out there, I’m sure you can find teams that are working on things that could be taken over by services that are now managed, right? Like I’m sure banks they have the equivalent of S3s that they run themselves and healthcare companies. They're of course like – The spend on traditional data centers is still growing. Cloud is a faster growing portion but a smaller part of the pie. And so a lot of these services are replicated at a lot of these big companies and you have a lot of engineers that are working. A lot of engineers, product folks, like DevOps folks that are all dedicated to these things that aren't necessarily in my opinion super business critical anymore. They're business critical until you transition over, but I think that you'd make a very easy case to say that they should be transitioned.

[00:16:35] JM: So Yelp is in a pretty competitive market. There are all these other ranking systems like Google of course. Google has its places, or whatever, locations. And then you have things like Tripadvisor. Were there strategies that you tried to employ to make the website harder to scrape? Or just in this competitive dog fight, was there a role for engineering to play?

[00:17:10] MS: The opposite is true with Google, right? You want Google to crawl you, crawl you as much as possible. So you try to make their lives easier. And the funny thing is you just want to make sure Google's crawler doesn't take you down. For a website like Yelp it's like the scale of which Google has to crawl your site is quite large. And so that's a huge – But it's an interesting question of like how much infrastructure should you dedicate to allowing Google to crawl you faster? It ends up being an interesting business problem of like cost of infrastructure versus what you assume the value of that extra crawling is worth.

So in terms of like scraping protection, things like that, there were always people selling databases of Yelp listings or whatever. But in terms of the specifics of like how we – Any business doesn't want their data being taken out, taken out to the open market. So we do the basics of what you would expect a Yelp like site to do so that you don't waste a lot of infrastructure on folks that are trying to get your listings and then sell them out on the dark webs.

So in terms of specifics, I don't really have many, but there wasn't really a competitive pull. There was more like just abuse and like wasted infrastructure was more the focus of that. The efforts weren't really focused on like, “Oh, this gives us a competitive advantage over anyone else.” But I think that the core of like how Yelp is – The search is competitive ends up being very competitive and better than other players is that every day the executive team wakes up and thinks about how do we make the local experience better? That's what the team's dedicated to thinking about every day in and out. And I’ve just seen that like through my investments in startups, like that kind of focus, like every day that Google wakes up, they're worried about some other fire. They have so many different kind of fingers in the fire on different businesses. They just don't have the executive focus to think about local search every day. That's a differentiator for any businesses. Whenever you're doing a startup or you're

working on a project, the more things you can say no to and focus in on what your core value proposition is, the better your product is. And I think that ultimately is like what you see with products like Yelp is just a very, very focused product on local and not that many distractions. And so you can just continually refine and get better and improve your machine learning. And everything's looked through the lens of local Yelp. And so that's a very different kind of way of looking at things versus being at Google and being like how does this affect web search? And web search is such a broad, multi-bucketed problem. And I worked at Google prior to working at Yelp and saw like how powerful the like hammer of web search is, but it engulfs so many different areas. And so like if you can just focus in on one of those areas, you're going to do better than an executive team focused on everything.

[00:20:35] JM: You did mention something interesting I want to go a little bit deeper on. So you said there are engineering ways to make it easier for Google to scrape your entire website. What kinds of engineering resources can you devote to that?

[00:20:51] MS: Well, so like every website, at least my knowledge might be dated, but the webmaster console that Google lets you log into that you then verify that you own the domain. They give you a bunch of knobs as to like how much to crawl you and things like that. The tags that you put on your site and the – There's like a bunch of different – it gets very complicated when you have a multi-language site, like a multi-language site that's also in different countries. So the example is you're in Berlin. You're a U.S. user. So you want English reviews to show up first, but then you also want German reviews but be able to translate them. How do you serve that to Google? So Google serves up the right web page for a user that searches for that business in Berlin.

So it's getting Google to be able to crawl all the right pages and be tagged in the right way is a non-trivial problem and it takes a lot of engineering resources to get right. But in terms of the original question of like what are the things to think about with getting crawled more by Google? The main thing is the webmaster console and there's also – You can create an XML sitemap. I don't know if they've opted for a different type of sitemap now. They probably do it in JSON or something else now, but just exposing all of the new content to Google so Google

knows what to crawl. That was done at a very high scale at Yelp because there're so many photos and reviews and different pieces of content coming into Yelp and Yelp wants to tell Google what is new on the site so it's not crawling older stuff that might not be as important to crawl that day. So they want to get all the changes. So when you do your search on Google you're ending up on a relevant page that has that content that you search for.

[00:22:51] JM: As the company grew, what were the kinds of challenges you ran into in terms of engineering management?

[00:23:00] MS: Yeah. I'd say about the first eight years of me being the head of engineering was building the recruiting engine to be able to sustain the growth. And inevitably the folks that leave, you need to be able to sustain both those parts. And just we were based in San Francisco, and when I took over the engineering team I think I was like 26. And what I found was that it was very difficult to get engineers with more experience to join me. And also I'd come from Google, and so Google was very effective at blocking any folks that wanted to join Yelp. They would parachute in with much bigger offers and make sure that those engineers did not come to Yelp. So I was kind of left with how do I grow the engineering team as fast as I can with the most talented engineers that I could?

And so I ended up kind of focusing in on new grad recruiting, which at the time was very underserved. And so we really scaled a lot of the engineering team through a lot of new grad hiring. We supplemented that with a lot of senior engineers that we surrounded those teams with. And so I kind of saw the makeup being like you have a staff level or principal level engineers or tech lead, tech lead type person surrounded by four or five new grads that were very talented and hopefully would become senior staff engineers themselves over the next five years.

So I think if you're looking on LinkedIn you'll see a lot of a Yelp folks have been at the company for quite a long time, and I think that speaks to the culture that we built and the reputation we built of building leaders. And out of the management team that I had when I left, I'd say about 50% of those managers were new grads when they started. And then over the course of five to

ten years that folks were there they became managers and ultimately built the culture there. And so it's a really pragmatic kind culture and they're of course looking to hire. So if anyone listening is looking for a really strong engineering team that wants to do the right thing for the world, you should check out Yelp.

[00:25:30] JM: In 2011 the company IPO'd. Did the IPO have any effect on how engineering worked at the company?

[00:25:41] MS: Yeah. It's funny to look at it in contrast to all the Stack activity going out there. The IPO was such a huge day. But like it really is a financing event. It's like you're raising – At the time I think Yelp had only raised 32 million or 34 million dollars in total. It's funny to think about it when you compare to these gigantic rounds that are happening now. And so that day we raised I think a 100 million dollars in the public markets, went public.

The effect on engineering was funny. It was like we were so scared that there was going to be this onslaught of traffic. And so we built a bunch of circuit breakers for different modules on the homepage. And so we really wanted to make sure that the site would stay up. And so we just turned – Where you'd have a module where it's like what reviews are near you or any of these database lookups. We created kind of like toggle switches that we could update via config to be able to handle any spike in traffic from people looking at the content.

And at the time the engineering team was pretty small. And so we were just like just deathly afraid of the site going down on that day and so we just like kind of all pushed for all – As many of these projects that were kind of circuit breakery type projects to be able to cut off different parts of the site. And then I think we did the normal things like setting up a bunch of extra database replicas and all that and as safe away as we could.

But at the time I think it was pre when we were not on AWS. So there was no magic auto scaling group that could scale. So we were stuck with the machines that we had. And so it was really a game of how do we cut resources if things start getting more turbulent in terms of traffic and we couldn't handle it. But team did a great job. We didn't have any downtime that

day. I'd say that the IPO in general marks a big event for a company and that you kind of go from the minor leagues to the major leagues. And so the coolest part of that is that talent actually is easier to find when you're a public company because people have a sense that you're going to be around for a little bit and you're being transparent with your revenue and your financials every quarter. People get a sense of certainty. And there are actually a lot more people that want that kind of stability than want the startup up and down. And so that was a pleasant surprise to me.

[00:28:32] JM: How much did the data engineering side of things evolve throughout your time at the company?

[00:28:40] MS: Yeah. So data engineering was like probably the closest thing to me because I was on the search side. I grew up through the search side of the world. And so when we started off at Yelp, everything was done with – When I got there, I remember coming from Google where I was working on MapReduce jobs and 10 terabytes of logs every day. Then I come into Yelp and the search log had 200,000 requests in it from all the searches that had happened. And 86,400 of those requests were load balancer uptime checks. So like every second the load balancer would check to see if the search engine was still up. And so that was half the search request roughly for the day.

And so I was not working with big data at that point. So my big data skills were put on hold for I'd say five years as we just built the nuts and bolts of like allowing people to find businesses that actually existed on Yelp and make sure they show up in search. Just the core like things that you think you get for free, like we had to build all that stuff and get it working reliably, get the indexing working.

So I think the biggest – The data infrastructure kind of started to evolve I'd say, we got to big data scale around year five. The data was being pushed into S3. We had built mrjob, which was like a MapReduce implementation in Python. Became a very popular open source library. That was kind of the beginnings of like the search service and our spell check service. We're

using big data. And then the ads the ad system started to use big data. That was kind of the beginnings of a lot of the data infrastructure.

Then we started to work on the Kafka pipelines. That started to emerge in like the 2014 kind of, 2015 kind of time range where the business started to need data in kind of near real-time moments. And really it was the evolution of – The biggest evolution was the evolution from a monolith. Like we had the Yelp main monolith that was a big chunk of code and that's where everyone developed on. And we got to about 150 engineers. And the push, the amount of work that was going into keeping the pushes, we had a couple of time a day – We had a morning push and an afternoon push. And just the amount of like backup that was occurring on these pushes and the amount of effort that was going into the infrastructure around the pushes was just becoming obvious that we needed to start moving to services.

And so the effort around starting to spawn into services created a need for data infrastructure for all the logs that were coming out of the different services to like upgrade. So that was all going to Kafka. So ultimately Kafka replaced a system. I think at the time we were using – There was a Facebook open source project for logging that we were using. And so we then upgraded to our own Kafka system.

And I'd say on the data warehouse side we started with just MySQL and S3, S3 type processing. And then as we evolved we moved – Some of the data teams started using the Databricks Spark open source system. And we used a lot of Redshift for the columnar databases for a lot of the different teams for being able to process all of the log data that was coming out of the system. And so I'd say it was a mix of that at the time that I was leaving. And we also used Splunk. Splunk was a very, very powerful tool for doing debugging and being able to like set alerts on different events that occurred. And so we had probably thousands of Splunk alerts that different teams used for like, "Oh, this food delivery order got stuck in this weird state," or different situations would occur and you'd want to alert an engineer to be able to like figure out what had happened in the system if something was in a weird state. And so that was occurring all throughout engineering. So I'd say Splunk was a big key and Redshift

was another key and then Spark was another part of the evolution of data infrastructure. There was a lot of custom stuff that was built as well. I think that that covers most of it hopefully.

[00:33:40] JM: Can you tell me more about it? So I mean you touched a little bit with the Splunk and the alerting mention there on some of the operational challenges or some of the operational themes of what you built at Yelp. Tell me more about how operations worked at Yelp and alerting and just the general DevOps practices.

[00:34:05] MS: Yeah. So at the beginning of Yelp we had a more traditional system administrator run ops team. And so we had like 10 engineers. And then when we needed to release something we'd kind of work with our systems team and do it in more of a traditional – The traditional way. And at the time there wasn't containerization. We didn't have containerization or any Terraform or any of the powerful tools that we have now. And so it was all very bespoke, “Oh, we need a new –” There's a package that we didn't have. We had on dev but we didn't have on stage or production. And so someone would have to go spend their day trying to figure out the package dependencies on the different machines. So that was the team that we started with. And then as the team got bigger and bigger, having a systems team or sys admin team as like a bottleneck to an ever-growing engineering team was like not functional. And so we hit a breaking point.

As we got into the into the 2012 when Yelp was transitioning from like the proprietary data center to AWS, that became a like an unblock of engineers could ask for machines. And before they would have to send in requests to a sys admin team that was kind of in control of how many machines were getting racked and stacked and connected. And so that kind of forced a conversation around how we wanted to kind of manage different systems and teams and the machines and then ultimately the on-call responsibilities for those different systems. And so really it was like the explosion of, “Okay. We now have over 150 engineers and we're now separating into services. And then is this kind of centralized systems team the right way to be managing all of these different services?” And the answer was no. And so what we did was we started to break apart those responsibilities. We created DevOps. We created ops deputies

that had power, had like controls over being able to operate in the production environment and then would be trained by this original systems team that we converted into DevOps.

And so the transition there was a big one and it took a complete reframe and reset with the engineering team of, “Hey, the days of pushing your service out or your code out and not being responsible for it are over. And we're now going to have an on-call rotation for every team. If a service goes down, you might be pinged at night about that, about that service.” And then that brings up a lot of other things that have to be updated around the organization and a lot of that comes down to like tooling and introspection. So like when something breaks, how do you figure out which service in the chain of services was broken so that the DevOps person that's on call can page the right teams or get everyone involved that needs to be involved? That is no small feat and those tools are not easy to build.

And so I think that the biggest reframe that we had from going from startup operations team to at-scale DevOps was this change from what this notion of having a firefighting team. I think in a lot of startups there's this firefighting culture where you're doing late nights. You get a page. When there's a problem, you just get into the cycle of, “Okay, we're going to solve this problem because you don't have the resources to solve the long-term structural problem.” You need to get the thing back up. And then what happens is a lot of credit goes to that firefighter for like keeping the site up, which they should get because they did something like really – They stayed up till midnight fixing some machine or getting something working. But it really ignores the like long-term vision which is fire prevention.

And so what you really want is how do we clear the forest of all the kindling so something doesn't ignite? And so how do you make that transition? And it's a combination of the right management hiring the right people. Getting enough breathing room for those firefighters to actually breathe and think about some long-term stuff. And a lot of times engineers won't even know that they're stuck in firefighting mode to be able to see the forest from the trees, right? And so that was a brutal, very, very tough set of years for the operations DevOps team in that 2012 to 2014 range. It was a very big reset and a very like bumpy transition. And I think every company that I’ve seen goes through that. It's always a challenge in terms of distributing the

responsibility to ops. I think if you're starting a company today it's a very different animal. But if you're transitioning an existing set of groups to this kind of DevOps mentality or organization to DevOps, it's no small feat.

[00:40:02] JM: Were you involved in any acquisitions when you were at Yelp? Were you in charge of merging any acquired infrastructure with the company?

[00:40:11] MS: Yeah. So I had to do that about four different times. The first of which was a company called Qype in Germany. They were a Yelp competitor focused on Europe mainly with traffic in Germany. And we had to transition all of their traffic. We ended up deciding to transition all of their traffic to Yelp, Yelp pages. And that was really an SEO magic trick of being able to transition traffic. Their site was getting millions of hits in Europe and we didn't want to maintain all of these different – They had quite landing pages very much like Yelp business page landing pages and we wanted to transition the reviews and users and businesses to Yelp and preserve the traffic associated with each of those pages. And so you can imagine that that transition was not an easy one. And simultaneously you have to manage the infrastructure that's handling the traffic currently. And so you have kind of two problems going on at once. And so that brought with it its own set of changes.

Luckily we had like a timeline for when we would shut down the infrastructure in Germany once the traffic had been transitioned safely. I'll talk about the hardest and one of the easier acquisitions that we had. The hardest was when we bought E24, which was a food delivery app in the early days in the food delivery space. It was an app built with a PHP infrastructure. It was written without many tests. And the kind of underlying infrastructure, they were pushing lots and lots of revenue through the system. And then once Yelp acquired the asset, it's like you're then responsible for the security and safety and infrastructure for this the site. And it was just like a mad dash to like get infrastructure engineers from Yelp to like be able to support and help. And we didn't have a lot of experience underpinning PHP systems and you kind of have to – You not only acquire this company and you have to – It's growing at some crazy rate, right? It was growing at like probably over 100%. It's like you have this crazy traffic growth. You also need to like take a system that was built by 10 or 11 people and then you have to

transition that to new infrastructure that's scaling. It's a non-trivial problem to handle and also transition it to the Yelp infrastructure stack of how we serve images and serve cookies, and all of that stuff has to change over time. The request from product of course ends up being, “Oh, we want to use the Yelp login credentials. We want to embed E24 all over. Get it more tightly coupled with the yelp backend.” That just ends up adding a lot of – It presents a lot of security and operational challenges that you have to – At the end of the day it comes back to what I was originally talking about with recruiting. It's like you need to be able to recruit a pipeline of engineers that can take over these challenges. And you run into staffing challenges where you get another staff level engineer. Where do you put that engineer? If you have a core business that's doing billions of revenue and then you have a new business that's doing tens of millions of revenue, like how do you decide where that engineer goes? And that's the challenge of engineering leaders is figuring out what the right balance is to like make sure you're not starving this new acquisition team that might end up becoming the core of your future business or like going to the traditional revenue stream. And this is a problem well understood in business. It's the innovator's dilemma. But it's a real challenge with staffing and engineering talent given how hard it is to hire engineers all over the world.

[00:44:24] JM: So you eventually left the company. Now you're doing investing. So you've seen what it's like to be inside one of these hyper-growth companies and the infrastructure problems that they're having and now you have a wider lens since you're seeing a lot of startups. What are the biggest markets in infrastructure and developer tooling that you're looking at? Are there any particular areas that you're focused on?

[00:44:51] MS: It's funny. Like it doesn't take a magician to figure out what the tools of the fortune 500 will be useful. What will the next infrastructure SaaS companies look like for the fortune 500? The reason is that if you work at any of these tech-forward companies where you're working around awesome engineers, working on the newest technologies that we've just been talking through the last 30 minutes. It gives you a cheat code as to like where companies that aren't able to hire those engineers will need to be over the next 10 to 20 years.

When you look at government and you look at healthcare and you look at finance, these folks are just now talking about transitioning to the cloud. They're talking about just now getting digital signatures or not needing to use fax machines. I always tell any entrepreneur that asks me what to work on next. I tell them to look for paper. Any company or process that needs to use paper is like a target for a startup, right? And so you can go into healthcare, and I think even to this day, I think to get my prescription, my doctor has to like fax. I think they now have an e-prescription system, but for a while they were having to like fax the Walgreens to like update the prescription. It's like you hear about stuff like that. Or you go into legal, all the legal work. It's all done with paper and WebEx signatures and all the stuff. We really have a cheat code from being in tech companies to seeing where the world's going to transition. It's just like looking at the stars. When you look at the stars at night, you're seeing a history. You're not seeing what's currently happening in the universe.

And so when you're at one of these forward-technology companies you're seeing where the world's going to be in 20 years. And so my job is just to translate that into which are the big pieces of infrastructure that people are going to be willing to pay for. It's actually not that hard when you look at it through that lens. And so that's what I've been doing for the last three years. I also have been working most recently on an app to help beginner surfers learn how to surf, the core skill being how to learn how to read waves. And so that app just got released on Friday. It's called SurfIQ. It's in the iOS App Store. Sorry android users. Soon. But that's my first foray into building something again.

I think ultimately at the end of Yelp I was pretty exhausted and needed a chance to kind of just collect my thoughts and see what I wanted to do next. And I've been passionate about surfing for the last six years. And so this is kind of my first foray into building something new. And I I've really enjoyed it and I'm not trying to make it a venture back thing. I'm just trying to enjoy life.

[00:47:59] JM: Cool. Well, Michael, thank you so much for coming on the show and sharing your reflections on Yelp as well as what you're doing today.

[00:48:07] MS: Awesome. And if anyone has any companies that they're raising for or anything on the show, feel free to reach out to me. I'm on Twitter at @stopman.

[END]