EPISODE 465

[INTRODUCTION]

[0:00:00.3] JM: There is a quote from Jeff Bezos, “70% of the work of building a business today is undifferentiated heavy-lifting. Only 30% is creative work. Things will be more exciting when those numbers are inverted.” That quote is from 2006 before Amazon Web Services had built most of their managed services.

In 2006, you had no choice but to manage your own database, your own data warehouse, your own search cluster. If your server crashed in the middle of the night, you had to wake up and fix it, and you had to deal with those engineering problems in addition to building your business. Technology today evolves much faster than in 2006. That’s partly because managed cloud services make operating a software company so much smoother. You can build faster, you can iterate faster, and there are fewer outages.

If you are an insurance company or a t-shirt manufacturing company, or an online education platform, software engineering is undifferentiated heavy-lifting. Your customers are not paying you for your expertise in databases, or your ability to configure load balancers. As a business, you should be focused on what the customers are paying you for and spending the minimal amount of time on rebuilding software that is available as a commodity cloud service.

Rich Archbold is the Director of Engineering at Intercom, a rapidly growing software company that allows for communication between customers and businesses. At Intercom, the engineering teams have adopted a philosophy called, “Run less software.” Running less software means reducing choices among engineering teams, and standardizing on technologies wherever possible.

When Intercom was in its early days, the systems were more heterogeneous. Different teams could choose whatever relational database they wanted; MySQL or PostgreSQL. They could choose whatever key value story they were most comfortable with.

The downside of all of this choice was that engineers who moved from one team to another might not know how to use the tools at the new team that they were moving to. If your previous team used MySQL and the new one uses PostsgreSQL, well you're going to have to figure out how to onboard with PostgreSQL.

That onboarding process is time that's not spent impacting the business directly. By reducing the number of different choices that engineering teams have and opting for managed services wherever possible, Intercom ships code at an extremely fast-pace, with very few outages.

In our conversation, Rich contrasts his experience at Intercom with his experiences working at Amazon Web Services and Facebook. Amazon and Facebook were built in a time when there was not a wealth of managed services to choose from. This discussion was a reminder of how much software engineering has changed because of cloud computing and the managed services built on top of raw cloud computing.

To learn more about Intercom, you can check out the fantastic Inside Intercom Podcast. It's a great window into how Intercom works and how they think about building products.

[SPONSOR MESSAGE]

[0:03:23.9] JM: Auth0 makes authentication easy. As a developer, you love building things that are fun, and authentication is not fun. Authentication is a pain. It can take hours to implement, and even once you have authentication, you have to keep all of your authentication code up to date.

Auth0 is the easiest and fastest way to implement real-world authentication and authorization architectures into your apps and APIs. Allow your users to login however you want; regular username and password, Facebook, Twitter, enterprise identity providers like AD and Office 365, or you can just let them login without passwords using an e-mail login like Slack, or phone login like WhatsApp.

Getting started is easy. You just grab the Auth0 STK for any platform that you need and you add a few lines of code to your project. Whether you're building a mobile app, a website or an API, they all need authentication.

Sign up for Auth0, that's the number 0, and get a free plan, or try the enterprise plan for 21 days at auth0.io/sedailly. That's A-U-T-H.io/sedaily. There's no credit card required and Auth0 is trusted by developers at Atlassian and Mozilla and The Wall Street Journal and many other companies who use authentication.

Simplify your authentication today and try it out at A-U-T-H.io/sedaily. Stop struggling with authentication and get back to building your core features with Auth0.

[INTERVIEW]

[0:05:14.1] JM: Rich Archbold is the Director of Engineering at Intercom. Rich, welcome to Software Engineering Daily.

[0:05:19.7] RA: Thanks very much. It's great to be here.

[0:05:20.6] JM: Today we're going to talk about a talk that you have given at several conferences, which is called Run Less Software. We're going to talk about it in the context of Intercom, which is a very popular software product, and also some of the experiences that you've had in prior at tech companies.

Since the focus is Intercom, and Intercom is a really detailed product, let's start up by giving people a perspective for what Intercom is. Could you just describe the Intercom product and how it manifests for users?

[0:05:55.6] RA: Sure. Intercom is a messaging platform that helps businesses connect with their customers. You can think of Intercom kind of like Slack or WhatsApp, only for businesses to use to help them have conversations with their customers.

Intercom's mission, like Intercom is very much a mission-driven company, and our mission is to make business personal. We think that a lot of the ways that businesses actually communicate with our customers today is like old, janky, stiff, spammy status quo. We would love to give customers the ability to – or we would love to give companies the ability to have really meaningful conversations with their customers and help them build real kind of relationships with them.

[0:06:45.5] JM: For people who have not seen Intercom or are not sure if they've seen Intercom, sometimes if you go to a webpage and you see a little chat icon in the lower right that is on the landing page interface, you can click on that and you can interface with people directly at the company. Oftentimes that is Intercom.

There are different companies that provide that, but Intercom provides I think the most popular one, or in my opinion, the one with the best UI. So how would you describe the engineering culture at Intercom?

[0:07:15.9] RA: That's a really interesting question. I would describe it as very product-first engineering culture, if that makes sense. It's not engineering-for-engineering sake. It's not fond of a lot of buzzwords, or not fond of things like we're going to service-oriented architecture because that is what everybody says you should do.

Everything is very much grounded in what do customers need? What do we need to do in order to build the best product for our customers? How do we think about identifying all of the most important problems? How do we make sure that we understand all of those problems from first principles, before starting to write code?

I think we spend an awful lot more time understanding the why and the what, before actually trying to build code or solutions. One of the other things that I think is really interesting about Intercom's engineering culture, again in that product, product-first, product thinking way is we have a very high amount of collaboration with all of the other kind of dependent disciplines involved in software development.

We worked very, very hand and glove with our product management teams, with our research teams, with our design teams, analytics teams. Everybody sits together in the one pod, everybody goes the same stand-up. I guess, that product-first, product thinking is probably the thing I think that shines through the most.

Probably the next most interesting element of the culture is the pace at which we move. I know like a lot of startups move quickly. At Facebook, we used to deploy code, new versions of Facebook twice today. I thought this was really fast. I thought it was fantastic. I came to Intercom and found we deployed new code 72 100 times a day at Facebook, at code push, took an hour to two hours at Intercom.

It takes 10 to 15 minutes fully automated push on green almost every single merged master results in its own deployed production. So that pace of development, that philosophy of shipping as your company's heartbeat, that is really interesting to me and it is probably one of the one of the things I'm most proud of about Intercom's culture, which has stayed very strong and ever present over the six and a half years or so that we've actually been around.

As you can imagine, as you get bigger and bigger and bigger, and as you add more and more people, and as you add more and more lines of code, the obvious thing to happen is that your software development process will slowdown, you will deploy less often, you will become more cautious.

I'm really proud of the fact that we have been able to stay fast and stay fast and stay fast, and also still achieve really good stability and have really good availability, really good scalability, really good efficiency while after doing that.

[0:10:22.0] JM: Yeah. Well, I think there's a direct connection between the fact that you're product focused company, but you also emphasize choosing standard technologies. You even go as far as – sometimes refer to it as boring technologies.

That's partially, because the boring technologies these days are really, really good. If you look at the buffet of options that you have available on AWS, you've got a ton of stuff that you can pick from and you don’t really need to reinvent the wheel most the time. You’re already introducing enough complexity building your own product for your customers.

You don’t really need to introduce more complexity by getting fancy if all the ingredients that you need for your recipe are already there. There’s actually Jeff Bezos quote that you liked to reference, “70% of building a business is undifferentiated heavy-lifting, and 30% is creative work.”

Things will be more exciting when those numbers are inverted. We’d obviously like to get to a place where 70% of building a business is fun creative work and only 30% is the building – piecing the building blocks together. How does that quote reflect your thinking at Intercom?

[0:11:39.4] RA: I think it comes very much back to this to this product-first, customer-first thinking. Any time we sit down to think about a problem we are going to try and solve for customers or feature, we are going to try and build.

Anytime we use customer language, we are we are never using MySQL or PostgreSQL, or MongoDB, or DynamoDB. We are talking about things like reliability, trust, privacy and functionality and solving problems.

That actually language really leads you to realize that database technology is not your competitive advantage, whether you choose PostgreSQL, or MySQL, or MongoDB, or DynamoDB is really irrelevant to your customers.

What is actually relevant to them is how fast you can get really good products to them, and how quickly you can evolve it as you learn how to make it better once it’s in their hands. That has really led down the path of using the standard technologies and moving towards this philosophy of Run Less Software, which is really all about saving time, moving faster and enabling us to build better software build and maintain better software more quickly, cheaply and easily.

[0:13:09.7] JM: You gave the specific example of being able to ship something like 70 times a day. How does picking standard “boring” software – how does that lead to a faster release schedule?

[0:13:25.3] RA: I guess what it has allowed us to do is any time somebody is trying to build a feature or figure out how to evolve a feature, a bunch of decisions have almost been made for them more easily. If this is a feature which is going to require a database, it is absolutely going to be built on top of AWS, Aurora MySQL. If it's a feature which requires a key value store, it's absolutely going to be built on top of AWS, Dynamo DB.

A lot of the standard building blocks that decisions are already made for you, and because they are already been – and because they are already made for you, the odds are you have been using these technologies over and over again. They are probably really well supported with libraries, monitoring, documentation, everything inside of Intercom, so that they are really easy to use, really easy to understand, and therefore easy to make new software with quickly.

I guess, one of the other things about it is we talk a lot in the industry these days about how talent is scarce. I read this fantastic – I think it was US government study, that showed that there were three times more software jobs advertised than there were hires made over the past year.

Engineering resources, like engineers it is just a war for talent out there. It's really hard to hire people. It's really hard to retain people. Everybody has open positions and everybody is actually trying to make the most out of the software engineers they have.

We feel by using all of the standard technologies that are primarily outsourced to a wonderfully trustable company like AWS who is innovating so fast, we actually get to run this really big scaling very, very fast backend infrastructure with a relatively few amount of engineers, which means we can have more and more engineers focused on our customer facing challenges building new products and features.

That also allows us to ship more quickly, I guess relative to if we have to spend more of our precious engineering time doing database upgrades, or figuring out how to monitor or instrument; loads of different backend technologies.

[SPONSOR MESSAGE]

[0:16:06.8] JM: At Software Engineering Daily, we need to keep our metrics reliable. If a botnet started listening to all of our episodes and we have nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot or a real user. That's why we use Incapsula, to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Incapsula checks that request before it reaches our servers and it filters the bot traffic preventing it from ever reaching us. Botnets and DDoS attacks are not just a threat to podcasts, they can impact your application too. Incapsula can protect API servers and micro-services from responding to unwanted requests.

To try Incapsula for yourself, go to Incapsula.com/2017podcasts and get a free enterprise trial of Incapsula. Incapsula's API gives you control over the security and performance of your application and that's true whether you have a complex micro-services architecture or a Wordpress site, like Software Engineering Daily.

Incapsula has a global network of over 30 data centers that optimize routing and cashier content. The same network of data centers are filtering your content for attackers and they're operating as a CDN and they're speeding up your application, but doing all of these for you and you can try it today for free by going to incapsula.com/2017podcasts and you can get that free enterprise trial of Incapsula. That's Incapsula.com/2017podcasts. Check it out. Thanks again, Incapsula.

[INTERVIEW CONTINUED]

[0:17:55.4] JM: Let's take a specific example. You referred to this just now. There was a time when you were running your own Mongo instance and you eventually moved Mongo to entirely AWS Aurora and DynamoDB to scale the user storage. What were the maintenance issues of running your own Mongo and what were the use cases within Mongo where some of them went to AWS Aurora, some of them went to DynamoDB?

[0:18:24.7] RA: Great question. One of the, I guess USP features of Intercom is that we allow you to store almost an infinite amount of schema-less data about your users on our platform.

Anybody who starts using Intercom, whether you are free or a paid user and start sending us thousands of things that – things that we call custom data attributes, and you can store integers, strings, arrays, whatever you want, you can change schema types seamlessly without doing anything, without action telling us anything.

This is a really, really powerful feature for customers to use, but it's also a very hard feature for us to maintain and scale on the backend. When we started building Intercom, we had this feature running on a single MongoDB replica set as we scaled, we started to do manual sharding of our MongoDB replica sets.
MongoDB has a max index size of – you can index a max of, I think it was like 64 columns or such. Within their indexes, we found TCP connection, pooling and scaling. Difficult to deal with, so we started to have tens of thousands of TCP connections, opening and closing to our MongoDB replica sets.

Every time we deployed new code, we were seeing lots of CPU churn with this. We were seeing disk space issues. We were seeing expensive queries threatening our database. It was expensive to run. We were just having lots and lots of problems with it. We had also started to use to Dvara is a TCP proxy. Dvara is a technology that was developed by Parse. It's written in Go. This was this extra complexity we now had to deploy in front of MongoDB.

All of this was probably leading to some of our most – some of our most difficult scaling challenges. Our teams were getting paged quite a lot at night, and it was just a pretty tough situation to be in, probably about 18 months ago let's say.

[0:20:51.7] JM: What about the selection of technologies to replace Mongo, how did you come to that conclusion?

[0:20:58.5] RA: How we came to that one was I think – I think it was really great. I think it was Intercom taking a product-first approach and really trying to understand our problem from first principles.

Our CTO stepped back from the chaos of all of the firefighting and took a long look at the stack and the features and everything that had evolved on the system that we had today and decided, do we to keep it the way it is? Are all of these features still relevant to customers first? Is there a way that we can simplify the problem by maybe coding features or coding scope?

We did a tiny little bit of that. What really decided that core of everything that we were providing was important and we needed to find a way to scale it appropriately and find the right cost and the r0ight type of operability for the solution.

We looked at a bunch of different ways in which we might solve this problem. I guess first we looked at MongoDB, because that was the one we were on and tried to figure out if we went to a natively sharded MongoDB, whether or not this would solve our problems. We reckoned it might, but we didn't think MongoDB was ever going to run natively on AWS as an AWS service.

By now, we had a really, really strong preference to using AWS, like native AWS infrastructures and service or platform as service. While we thought it was possible to solve it with MongoDB, we would much rather using an AWS service.

Our next favorite AWS service is probably AWS Aurora. We tried modeling all of our user service on AWS Aurora. Aurora like any – like any MySQL database still only has a single host scaling point. We reckoned, while we could do it all on MySQL, we would eventually run into scaling bottlenecks and need to shard our MySQL cluster and sharding MySQL wasn't something we really, really wanted to do so early in the lifecycle of this new iteration of a solution.

Then we tried on our next favorite AWS solution, which is AWS Dynamo DB, and we tried modeling all of the data there. That had nice schema-less properties that I use to add and remove new custom data attribute columns really easily. It was natively scalable, it was cheap. But the problem with DynamoDB was it didn't support multi-value primary keys. Each element of customer's customer data has a three-part primary key, which is the Apple ID, customer ID and e-mail address.

We would've had to do some crazy concatenated string primary key, and so we decided that wouldn't work. But then one of the ways we think about choosing standard technologies and we think part of the art and magic to it is being able to take a complex problem and break it down into pieces, which look like they would start to fit into some of your standard building blocks.

We were able to separate the problem of the multi-part primary keys and model that nicely in Aurora. We were able to take out the part which is the schema-less data and put that in DynamoDB and write a little bit of Ruby that would help us knit both of those two things together in a higher-level library. We got this lovely solution, which allows us to use the best of both worlds and allows us to use Aurora for the thing it does best, and Dynamo for the thing it does best.

That has been fantastic for us. We've seen way, way, way better availability. We are still in the process of fully rolling it out where out for a significant portion of our apps, but so far we are expecting somewhere in the order of an overall 90% cost reduction. We've had no availability issues. We now a solution, which is natively scalable and it is basically allowing us to move from something which was about 3 to 4 engineers worth of like constant operation and constant worry and for data that was sitting outside of our VPC to data that is now a 100% inside of our VPC, a 100% on native AWS services where all of the main bulky undifferentiated operations tasks are now handled by AWS for us.

[0:25:44.6] JM: No more of the Mongo maintenance issues, thankfully.

[0:25:48.7] RA: Hopefully. Hopefully. I don't want to give out about MongoDB. I think it's an absolutely fantastic piece of software. I think there's there's absolutely no way Intercom would be where it was today without MongoDB. I think this thing about choosing standard technologies is really interesting and opinionated.

When I give the talk, one of the things I talk about is how Intercom's standard technologies are Intercoms, and they fit us and they fit our experience, and they fit our engineers. But MongoDB could be perfect standard technology for a company who has a slightly different use case to us.

[0:26:27.9] JM: Right. Makes sense. Another example, you've talked about consolidating all the relational databases at Intercom to Amazon RDS is another instance of standardizing on Amazon technology. But before that, your effort was actually divided between administration of two flavors of AWS RDS. You had MySQL and PostgreSQL. These are both relational databases, seems standardized enough. Why did you need to move everything to MySQL? Why did you have to get rid of the PostgreSQL, AWS RDS?

[0:27:05.1] RA: That is an interesting question. I think it's even more interesting to ask why did we end up with MySQL and PostgreSQL to begin with.

[0:27:12.6] JM: Okay. Sure.

[0:27:15.1] RA: I don't think there is a good answer to this, other than we didn't have a standard. We were a fast-moving startup who had hired a bunch of people who had preferences and experience with a bunch of different technologies. People were like building new features and using whatever they were comfortable with. We ended up with a bunch of – with most of our features built on top of MySQL on a bunch brother features built on top of PostgreSQL.

We ended up with probably 90% of our engineers really comfortable with MySQL, and a very small number of our engineers comfortable administering PostgreSQL databases. We had had a couple of like annoying outages, short outages where one of our PostgreSQL databases had had failed and maybe we hadn't noticed it, because the majority of our operational tooling was built around MySQL.

While it wasn't as huge pain for us, it was this constant little niggle of outages, or expensive operations where you are trying to figure out how to cleanly change a setting, or deal with some performance tuning on PostgreSQL.

We had this offsite probably about six months after I joined the team, where we were trying to figure out how can we save time? How can how can we make things run a little bit better, a little bit easier remove a bunch of cognitive load? This is where this Run Less Software idea originated from.

I guess, the most immediate low-hanging fruit that was uncontroversial was let's take the time now and invest and move away from PostgreSQL to the thing that we know and love; MySQL. That further incentivizes then to let's actually start bringing in some MySQL training for everybody else in the company, so maybe anybody who wasn't as familiar with MySQL.

We brought in a bunch of MySQL consultants and got them to do a bunch of training with us, in order to help upscale anybody who is maybe would've preferred PostgreSQL and now feels a little bit more forced into using MySQL.

[0:29:38.8] JM: Sounds great. There was an example you gave where you could not outsource the technology. That was the example of elastic search. You have to manage your own elastic search on bare metal. Why is that?

[0:29:56.2] RA: Why is that? We started using elastic search when it was still a relatively young technology. There wasn't anybody who was doing outsourced elastic search at the time, and that is why we started doing it.

As we started to do it, we invested in a bunch of training, we got some elastic search consultancy, we sent some people over to Amsterdam to some – or to do some elastic search training courses, and we decided if this is something we are going to have to run ourselves, we are going to do it the right way and learn about it and become as close to experts as we can in it. I guess, that was the reason why we started doing it.

Over time, a bunch of different companies have brought out - hosted elastic search products, Elastico have their own; hosted elastic search product. AWS has one as well. I think there's probably some other companies who do it now as well.

[0:30:58.4] JM: You were just too early.

[0:30:59.5] RA: Yes, we were too early. But about every six months, we revisit this and decide, "Hey, is now the time for us to try and outsource elastic search?" The honest truth is the AWS product I believe is nowhere near good enough and nowhere near stable enough for somebody who runs an elastic search cluster of our size.

Some of the other elastic search providers, in order to use them it would require us sending customer data outside of our VPC, which is something I'm super uncomfortable doing with from a security and privacy and GDP perspective.

[0:31:36.4] JM: Don't you have to do that, if you're storing customer data in AWS databases?

[0:31:42.4] RA: You do to a certain extent, but AWS have so much certification and so much security that it is still a very easy and tight and safe security story that we can tell ourselves and tell our customers. Having worked inside of AWS myself for nearly 8 years, I know the lengths to which the lengths and maturity to which AWS goes in order to safeguard all of its customer data.

I guess, I'm a little bit of an AWS fanboy having worked there for so long, and I really, really believe in how well the company thinks about security and thinks about privacy. I'm just willing to put an awful lot of trust in them, to be honest.

[0:32:32.1] JM: Yeah. Trust is hard-won and easily lost as Bezos likes to say. We're talking about all these managed services, and it makes me wonder, is cost ever a concern here when you're just going whole hog at all these AWS services?

[0:32:49.8] RA: It depends on what you mean by cost and what you mean by expense and what you think are your commodities and what your constraints are. I can tell you that from my perspective, the most things I worry about spending most or the things that I think are the most expensive to get and you need to be the most careful how you spend them is engineering time. Because I think that is your most precious resource.

I think money is relatively easy to get. Interest rates are at an all-time low. I think the stock market is high. I think investors are incentivized to put their money anywhere other than a bank. I think if you are a fast-growing, credible, successful startup, it's relatively easy to get money. I think it's easier to get money than it is to hire a bunch of really great software developers.

That's actually the biggest expense I think about. But also, I mean AWS over time consistently their prices go down rather than up. By betting on AWS over time, we automatically know things are bound to get cheaper.

Other things by using more of AWS, things also get cheaper the more you're willing to reserve or whatever upfront the more comfortable you are with them, the bigger discounts they have.

Everybody knows Amazon runs on very, very small margins relative to bigger companies, so it's hard to believe somebody else is going to be able to do it sustainably for the long term more cheaply than AWS.

I guess there's a bunch of opinions in there. I'm not going to talk too much company financials, but we're still healthy, so – Honestly, I think engineer time is the biggest cost you need to worry about, and engineer efficiency, rather than raw dollar cost. But even if you are wondering about and worrying about raw dollar cost, I still think investing in AWS for your undifferentiated heavy-lifting is still the right way to go.
[SPONSOR MESSAGE]

[0:35:14.0] JM: Do you have a product that is sold to software engineers? Are you looking to hire software engineers? Become a sponsor of Software Engineering Daily and support the show while getting your company into the ears of 24,000 developers around the world.

Developers listen to Software Engineering Daily to find out about the latest strategies and tools for building software. Send me an e-mail to find out more, jeff@softwareengineeringdaily.com. The sponsors of Software Engineering Daily make this show possible, and I have enjoyed advertising for some of the brands that I personally love using in my software projects.

If you're curious about becoming a sponsor, send me an e-mail, or e-mail your marketing director and tell them that they should send me an e-mail, jeff@softwareengineeringdaily.com.

Thanks as always for listening and supporting the show. Let's get on with the show.

[INTERVIEW CONTINUED]

[0:36:16.2] JM: Keeping the pace of the company as fast as possible – I mean, always the engineering time, cutting down engineering time spent on managing infrastructure is going to make the company go at a faster pace. You really emphasize this in at least in the instance of the talk that I saw, your sense of paranoia in how you think about competition. You actually led off your talk with this discussion of competition and how you could imagine a world in which Slack or Facebook come out with things that are competitive pressures against Intercom.

I can imagine that affects the pace at which you feel you need to sprint as a company; the messaging communications company. How does that affect your engineering management in your choice of software?

[0:37:09.6] RA: It affects it a lot and. One of the things I like to do is compare and contrast my time on Amazon, Facebook and Intercom into a couple of phrases, I think of my time at Amazon – when I was at Amazon, I think of – I describe Amazon as like high-quality, low-cost and kind of a frugal environment.

I describe Facebook as innovation-focused, fast-moving and generous towards their employees. I think of then Intercom as like product-focused, dizzyingly fast, crazily fast-moving and kind towards its employees. That dizzily fast-moving, the feeling that life is short, time is precious, opportunities are fleeting, now is the time you need to make the most of this opportunity is kind of really this feeling that permeates right the way throughout the company and leads to this feeling of, "Let's actually move as fast as we can. Let's be aware that we are on the wave of something. We are writing the quest, or we are riding the crest of this consumer communications revolution almost where companies want to talk to their customers like real people and build real relationships with them."

Intercom was the first to come out with an in-app messenger. We were the first ones to have a single integrated platform for sales, marketing, product management, customer supports to all talk to their customers in the same way using the same platform. We have this advantage, "Let's keep going, let's not actually get caught by any of our competitors."

If you think like that, you just really want to move as fast as you possibly can. You have all of these great ideas and you are a little bit afraid that your R&D capacity, your own ability to build software and build great software and learn from it and build some more, that is really your own competitor. You're actually competing against yourself.

You have the opportunity to win if you can only just keep moving fast enough. That feeling is definitely part of it. I think part of the other part of it for me is I didn't start out my life as a director of engineering. I started out my career as sys admin. Back 15 years ago when I was a sys admin, sys admin meant rocking servers, cabling them, imaging them, recompiling kernels, a bunch of really, really low- level work.

Overtime I've seen that work be abstracted, commoditized, eradicated I guess a little bit by AWS and infrastructures and service. This feeling of the industry is moving really, really fast around me, and a lot of things that were once valuable have now been commoditized and abstracted away and we need to keep – all of us need to be aware of that and keep moving up and up and up the stack and staying focused on what customers need, rather than what is technology of the day. We need to keep moving with that and rolling with that I guess.

[0:40:41.9] JM: You might know, I worked at Amazon briefly as well, so I saw that frugality firsthand, where the company just really keeps costs low. Why is Amazon so unique there? Why haven't other people instituted the frugality as a virtue? Did it just have to do with the fact that Amazon started when there were fewer options for companies to work at, and so they were able to say, "Look, we're not going to give you like massive free lunch. We're not going to give you tons of perks. We're going to give you a decent salary. We're going to give you inspiring vision." How do they do that? I'm sure you've reflected on that. I certainly have.

[0:41:20.0] RA: I think it's unique in one way. I think it's really unique and opinionated. I think people are always drawn to things that are different and opinionated and anti-status quo. I think, that is one thing it has going for it. I think the other thing it has going for it is, I think people follow leaders and people want to work for people who have a really great vision. I think people want to work for mission-driven companies, and I think Amazon has all of that in spades. I think Jeff Bezos is one of visionaries of our time.

Amazon is very values-driven company. I think it has a has a very clear mission, so I think that helps. I think there's a lot of ridiculously smart people there as well and the absolute best perk that you can ever have is working with world-class people. I think there's still a lot of amazingly world-class people at Amazon.

I think the last thing if I may be a little bit more practical is the stock price. Certainly, I know over the last four, five years, Amazon has had an average of I think 41% stock price growth over the last four or five years. It's fast approaching, being one of the world's first trillion dollar businesses. I think that the fact that a large part of Amazon's – of Amazon's employees compensation is stock-based or SU-based, and you can see this compounding 41% growth year after year. I think all of those things together provides a really, really compelling reason to work there or stay working there.

[0:43:08.5] JM: When you were working at AWS, did you did you have any ideas around this Run Less Software idea? I mean, when I think about working at Amazon, that doesn't really seem like – I can't recall anything similar to a principle like Run Less Software. But Amazon is a very different company.

[0:43:28.2] RA: I actually don't think we had it at Amazon, or certainly where I was inside of AWS. We were we were operating below the hypervisor. Us using services like AWS in order to build our own services would've been creating circular dependencies. That was certainly very much frowned upon. We were all about considering ourselves like a utility, understanding all of our single points of failure and making sure that we had absolutely full control over them. We as a team had full control over them.

I think that started to change over time though. I think that, because when I see things like DynamoDB having an outage, it seems like half of AWS goes down now. Or when S3 goes down, half of AWS goes down.

I'm thinking that maybe they started to have more of a change in their philosophy internally there where they are now starting to run less software for one of the better word and start to use more of their own technologies and just accept that when DynamoDB goes down, or when S3 goes down, it's effectively going to be an AWS region-wide outage.

The benefit of that is that each of the individual services who depend on Dynamo or S3 are going to become more available, because they don't have to run their own key value stores, they don't have to run their own MySQL databases or things like that. That would be my guess.

[0:45:06.6] JM: Does it worry you that so many people are on AWS at this point and it is – it's like a utility, it was like aa multinational utility, it's like a global utility, I mean, I sometimes am a little worried about it, especially when you have these outages and I know how trustworthy

Amazon is also, but these things just happen. You have long-tail events and they happen. It's scary. I guess, it's like the banking system though.

[0:45:33.7] RA: I think it is and it isn't scary. Part of the reasons why I think it isn't scary is that if you are in US East One, if you are based in US East One where I think half the internet is basically based at this stage, and there is a US East One outage, it's effectively like the internet is down.

Even though you are down, also all of your customers are down as well. Almost like people don't – You are down, because all of your customers are down. That's one reason that I'm not afraid of. But the other reason that I am afraid of it is that we provide customer communication software and customer messaging software.

The times when customers are going to want to talk to businesses or businesses are going to want to talk to customers is pretty accused in terms of outage. I really, really want to be able to find a way for Intercom to be able to operate in that environment. Whether it's moving out of US East One, or running across multiple regions, it's definitely something which is on my mind and I would love to figure out a way to fix over the coming 6 to 12 months.

[0:46:47.9] JM: When we're talking about running less software, how does that apply to, I don't want to say service-oriented architecture because you already poo-pooed that name a little bit earlier, but I imagine you do have some distributed set of services. Maybe it's on Kubernetes, maybe it's on Amazon ECS. How does that look? How does the service ownership and deployment process look?

[0:47:15.2] RA: How does it look? How do we think about that? We definitely do have a bunch of discrete services inside of Intercom. Maybe not when people think about service orientated architecture, I think about things being able to fail independently of each other or having really, really strong versioning, or not so much tight coupling between different versions of different services.

I don't think we are all the way towards service orientated architecture or microservices as the maybe Wikipedia definition of them describes. But we absolutely do have a bunch of distributed services. The way I think about run less software, the way it helps this is that there are common building blocks across all of these different services. We are doing prod-like data storage in the same way, we are doing key value data storage in the same way, we are doing caching in the same way. We are generally writing our software using the same – using the same language as we are absolutely monitoring it in the same way. We are deploying it in the same way.

What this means is that it makes it really easy for engineers to switch from one team to another and get up to speed really, really quickly. We actually try and have this run less software, maybe even use less processes, or use standard processes. I would go so far as to say each team would do planning in the same way, they would do planning at the same, they would do retrospectives in the same way, they would phrase all of their goals in the same way, they would write all of their roadmaps in the same way, they would use the same software, they would use the same building blocks.

It basically allows us to have this really, really agile team that if we decide our engaged product is the most important thing we need to focus on over the next six months that we can pretty easily move a bunch of our software developers from a team that were not on engage into engage and have them come up to speed quickly, have them be able to maintain and evolve that software stock really quickly, be able to understand the team's culture, understand the teams processes almost seamlessly.

[0:49:26.5] JM: I've done some shows with people who are building service proxying tools, like envoy. I talked to Matt Klein from Lyft, and envoy standardizes the communications between different services. It gives you a standardized way of doing load balancing. It gives you a standardized way of passing messages between different services.

An equally valid way of standardizing how the different teams at Intercom operate with each other, or just operate on their own is just to have cultural processes. It sounds like that's what you have seen more at Intercom in terms of standardization is like people just have a way of doing things, and that way of doing things has permeated whether or not it has permeated to the level of a standard micro service unit, like a docker container.

[0:50:20.9] RA: Yeah. I think that's very fair. I think maybe we are maybe more advanced on the standardization of process and thinking and maybe not as far along on the standardization of inter-service communication. Though I would say dropping a message into an SQSQ and having it picked up by another service to be acted on asynchronously is a pretty standard way of different services in Intercom communicating with each other.

But then, I think the things that slow you down are solving problems, people, cultural, teams forming, team dynamics, understanding all of your biggest problems are solved by multiple teams and multiple disciplines collaborating together. Definitely, I think that the standardization or the ways of working or ways of building that we do have definitely helped large groups of people come together to solve new problems that we haven't encountered before.

[0:50:20.9] JM: It's so interesting to talk you about this because you know a lot of the shows that I did think earlier were – I would say there's a several different types of companies I talk to. Companies like Netflix and Amazon and Facebook and Google, they were formed in a time where it was not as much you could take everything off the shelf and just kind of build the product development culture in an atmosphere where you can take these building blocks off the shelf.

But now I talk to companies like Intercom and talk to Fiverr a while ago. I've talked to a number of other – the companies that have grown quite large on – off the backs purely of cloud services. I guess, you could argue Netflix is like that at this point. Yes, they made such a aggressive move to the cloud.

Can you talk about some – are there any principles that come to mind that are true in this environment where you can take so much stuff off the shelf that are not necessarily true in the older world of software, where you really did have to build a lot? It almost seems like build versus buy, you buy almost all the time. Like just buy.

[0:52:46.6] RA: Yeah. Let me think about that one. I would say maybe the biggest difference that I see between how we operate now at Intercom versus how I would've say operated at Amazon – particularly Amazon, maybe less so than Facebook, one of the core values we have inside of our R&D org in Intercom is think big, start small.

We would think about a problem we want to solve. We would understand it from first principles. We would then think really big and design our – design our final solution, and then pair it right back and scope it right back to like the smallest, simplest, fastest thing that we can do that will solve problem – that will solve a real problem for customers, almost like pairing it back to your MPV; ship that and ship it to a small group of people and learn an iterate and iterate and iterate, and slowly but surely build towards that bigger thing.

I felt like when I was at Amazon, there was a think big, start big, rather than think big, start small. You had to design that V3, like design that really big all-encompassing solution. There was a lot more pressure for anything that kind of went to market or anything that you are going to ship into production had to be fully featured, fully finished incredibly reliable.

Maybe cost wasn't so much of an issue, but it had to be really, really well polished and really, really well thought out. It was much less of an opportunity, like everything was so critical. There was much less of an opportunity to learn or make mistakes or really move fast. The average project would take 6 to 9 months, or it wouldn't be uncommon for a project to take a year, or even more.

Whereas, now I guess in startup world, we can definitely do that think big, start small, move fast. Part of the reason you can move fast I guess is because you have all these building blocks. Whereas again, when I was in Amazon, you didn't have this money, those building blocks. You had to provision your own servers, you had to run your own MySQL database, you had to figure out your own caching infrastructure. Does that make sense?

[0:55:16.1] JM: It does. It does, completely. I know we're up against time. To close off, so in this atmosphere of running less software where you get to take a lot of cloud services off the shelf, I imagine that managing outages becomes less of a frequent issue. But since I saw you give this talk at SRE con, which is the site reliability engineering, maybe you could talk about just briefly how you manage outages, what your response team run outages or instant response looks like, and how that affects the culture at Intercom.

[0:55:51.4] RA: Great question. This is actually a lovely story, I think. When I started at Intercom three and a half years ago, we were averaging about a 99.91% overall system availability, which is still pretty good. 99.9 was our customer facing commitment. I used to record every outage we had. I would send out e-mails every week to the company giving them an update on our availability.

I would track all of the sources of power outages, and are outages were split between ones which were caused by infrastructure scaling events versus ones that were caused by us moving fast and us maybe deploying changes too quickly, or without properly understanding what we were doing.

Over time, it's been really interesting to see the different types of outages, the different measures of availability change. Over those intervening three years, we've outsourced more and more of our undifferentiated heavy lifting to AWS. We've learned more about how we develop software. We make very, very heavy use of feature flags, even though we are deploying code to production 100 times a day. Almost every change is going at behind the feature flag or certainly a lot of the major changes are going up behind feature flag, so it's almost like dark launches and you have the ability to turn things on and off very, very quickly.

Now were at a place where I think we're going to end this year with 99.97 availability, which is a significant increase from where we've been. I think this is going to be our best year for availability ever. Our biggest outages were maybe RA-related outages. We had very, very, very, very few real operational outages.

We still have, "Hey, software developer has deployed code, and it has had an unintended consequence and wasn't behind a feature flag." But even those are getting rarer and rarer. I'm an **[0:57:56.6]** guy. I love high availability, I love hard scalability challenges. I'm probably going to curse myself here, but things have been coming – things actually have been becoming really, really quiet these days. I harken back for the days when things were a little bit more exciting.

[0:58:14.5] JM: Be careful what you wish for.

[0:58:17.3] RA: Indeed.

[0:58:19.7] JM: Okay, Rich it's been great talking to you. I really enjoyed your talk. I really enjoyed this conversation. Time flew by. I'm looking forward to doing more shows about Intercom n the future.

[0:58:28.9] RA: Great. Thanks very much, Jeff.

[END OF INTERVIEW]

[0:58:33.2] JM: Heptio was founded by two creators of the Kubernetes Project, to support and advance the Open Kubernetes Ecosystem. Heptio unleashes the technology-driven enterprise with products that help customers realize the full potential of Kubernetes and transform IT into a business accelerator.

Heptio's products improve the overall experience and reduce the cost and complexity of running Kubernetes and related technologies in production environments. They build products, they build open source tools and they provide training and services and support that bring people closer to upstream Kubernetes.

You could find out how Heptio can make Kubernetes easier at heptio.com/sedaily. Joe Beda and Craig McLuckie who are the founders of Heptio have both been on Software Engineering Daily and they were fantastic when they came on. They're really fluent in what is going on in the Kubernetes ecosystem, because they helped build it.

To find out more about the company that they're building with Heptio and to find training and resources and products built for Kubernetes, go to heptio.com/sedaily.

Thanks to Heptio for being a sponsor of Software Engineering Daily. I'm really proud to have the company onboard.

[END]